An Abduction-based Method for Explaining Non-entailments of Semantic Matching

Ivan Gocev [a,*], Georgios Meditskos [a] and Nick Bassiliades [a]
[a] *School of Informatics, Aristotle University of Thessaloniki, Greece*
E-mails: ivangochev@csd.auth.gr, gmeditsk@csd.auth.gr, nbassili@csd.auth.gr

Abstract. Explainable Artificial Intelligence (XAI) attempts to give explanations for decisions made by AI systems. Despite the fact that for knowledge-based systems this is perceived as inherently easier than for black-box AI systems based on Machine Learning, research is still required for computing satisfactory explanations of reasoning results. In this paper, we focus on explaining non-entailments as a result of semantic matching in $\mathcal{EL}_\perp$ ontologies. In the cases where the result of semantic matching is an entailment, the already established methods of justifications and proofs provide excellent results. On the other hand, the cases in which the result of semantic matching presents in the form of a non-entailment demand an alternative approach. Inspired by abductive reasoning techniques, we present a method for computing subtree isomorphisms between graphical representations of $\mathcal{EL}_\perp$ concept descriptions, which are then used to construct solutions to abduction problems, i.e. explanations, for semantic matching non-entailments in $\mathcal{EL}_\perp$ ontologies. We then illustrate our method with an example scenario and discuss the results.

Keywords: Explanations, Description Logics, Semantic Matching, Non-entailments, Abduction

1. Introduction

Today, the vast amount of data that is available has driven research into Machine Learning (ML) and Deep Learning approaches, which provide AI systems with the possibility to autonomously learn and adapt to various scenarios. These models consist of black-box structures, i.e. they provide outcomes without revealing any information about their underlying workings. To this end, with knowledge based systems being inherently explainable, explanations techniques are more focused on black-box systems [1]. However, the information that these approaches provide is often numerical, which lacks context and needs additional information to understand how a result was achieved or to interpret that result. Knowledge based systems and reasoning are well suited to provide help here, by encoding information and adding context to it, as well as providing the possibility to reason with that context. In addition, explanation techniques in Symbolic AI could help, when integrated with Subsymbolic AI, to understand the underlying workings of Subsymbolic approaches. Furthermore, the existing research in explainability for knowledge base systems is not complete and has room for improvement. E.g., the already established methods of justifications [10] and proofs [14] provide excellent results when explaining logical inferences of knowledge bases. However, they fall short when explaining why observations are not a logical consequence of some knowledge base [15, 16, 38], when, in reality, they should be. Widely used approaches that are based on reasoning techniques in Symbolic AI, for example semantic matching, are enormously enhanced when explanations are introduced. Thus, research of explanation techniques for knowledge-based systems, even though perceived inherently easier than for black-box AI systems, is still a vibrant topic.

*Corresponding author. E-mail: ivangochev@csd.auth.gr.

Ontologies offer a suitable way to capture and classify domain knowledge from human experts, making them practical in various substantial fields. SNOMED CT[1] is a standardized medical ontology that provides codes and definitions for medical terms [2], and the ChEBI ontology[2] represents chemical entities of biological interest [3], with both including over 300,000 axioms. In addition, ontologies have been developed for various industrial fields such as electronics [4], energy [5], process engineering [6], and construction [7], to name a few. Besides modeling domain knowledge, ontologies offer numerous ways to support users' decision making [8, 9]. This is achieved by means of logical reasoning, which provides the possibility to perform reasoning tasks such as consistency checking, instance retrieval, and inference. To further enhance the decision-making support, methods that provide explanations for reasoning results are included in knowledge-based systems [10, 14], thus creating explainability within the scope of Symbolic AI.

Ontologies are often used to represent domain knowledge in the form of axioms. Based on these axioms, reasoners are then utilized to infer new knowledge (axioms) that are a direct consequence of the modeled ones. The inferred axioms are called entailments and they are a direct consequence of knowledge represented in ontologies. Naturally, methods to explain entailments are constructed, and are, in fact, quite thoroughly researched. E.g., an established method to explain entailments are justifications, which provide a minimal subset of axioms from an ontology sufficient enough for an entailment to hold [10]. Recently, proofs have also been used to explain entailments and present a step-by-step process of obtaining an entailment from a knowledge-base [12–14].

However, the question now asked is: how can we explain non-entailments? Non-entailments are axioms that do not logically follow from a knowledge base. Hence, if we were to explain non-entailments, we would not be able to directly derive from the existing knowledge the reasons for why an axiom is not entailed. This indicates that we need to identify new knowledge, i.e. "missing" from the ontology, relevant to the concepts in the non-entailment, in order to explain it. Considering this, a classical approach to explain non-entailments is *abduction*, which is used to generate hypotheses that contain "missing" knowledge, such that when added to the ontology the non-entailment becomes entailed. In recent years, accent has been put on abductive reasoning in ontologies and creating methods to explain non-entailments [15, 27, 29, 38].

Naturally, in abduction, a set of possible abducibles is determined [16, 29]. Such a set provides the possible concepts or axioms that could be abducted and it represents a form of a constraint on the hypothesis. Recently, homomorphisms have been used to generate abductive solutions for TBoxes [15], which do not explicitly constraint the abduction to a set of predetermined abducibles. Still, they capture entailments through axioms that contain atomic concepts only. Though they extend the set of common minimality criteria [27], such as subset, size, and semantic minimality, and filter hypotheses that do not carry meaningful information, homomorphisms restrict hypotheses to concepts only and exclude role restrictions in abductive solutions, thus dismissing other potential abductive solutions. For example, considering TBox abduction, abductive solutions could be restricted to only include axioms of a certain type, such as concept inclusions of the form $A \sqsubseteq B$, containing only atomic concepts. However, in reality, the abductive solution may not only consist of such axioms. The key reason why an observation is not entailed may be a role restriction on a certain concept, represented by an axiom consisting of role restrictions as well, such as $A \sqsubseteq \exists r.B$. If we omit role restrictions in hypotheses, then an abductive solution may not be found even if it does, in fact, exist. This challenge is emphasized in [16]. To resolve it, they generate predefined patterns based on justifications, which represent forms of axioms that can be abducted.

One use case that can benefit from explaining non-entailments is semantic matching. Within [18], semantic matchmaking (or semantic matching) and its most widely used degrees of matches are introduced, which are: exact, plugin, subsume and intersection. In the context of intersection based matching, [20] offers experiences and modeling guidelines to achieve a robust matchmaking service. In [21], the degrees of matches are applied to semantic descriptions that capture the service "as a whole", represented by a concept in a knowledge base. In [22], an approach for matching service descriptions based on WSMO framework is presented and in [23] a web service matchmaking algorithm is presented that can be used to process an initial set of candidate services in order to aid the search of more fine-grained discovery approaches. On the other hand, [24] present the notion of stateless services and an approach for the matching of services with high precision and recall, as well as prove that matching can be reduced to

[1] https://www.snomed.org/
[2] https://www.ebi.ac.uk/chebi/

conjunctive query equivalence w.r.t. TBox. This shows the wide use of semantic matching in Semantic Web Technologies. Thus, generating explanations for semantic matching could enhance scenarios that utilize this vastly used approach.

Explaining semantic matching helps make AI systems transparent and understandable. For example, in [42] semantic matching is used in an industrial scenario. At first ontologies are used to represent machine knowledge and semantic matching is used to match machines' capabilities with certain product requirements. Then, explanations for the outcomes of semantic matches are generated, which help users understand the reasons why a certain product cannot be produced. [46] presents a satisfiability-based approach to explain results of semantic matching and in [19, 47] we can see the utilization of semantic matching for an e-marketplace. To discover potential matches, they use abduction, which in a way explains why a negative result of a semantic match is obtained, by showing how the negative result can be converted into a positive result. In addition, semantic matching is widely known to be used in service discovery [22, 52].

Generating explanations for results of semantic matching in Description Logics (DLs) still poses interesting questions. The idea is to use semantic matching to identify whether concepts are, or are not, semantically related w.r.t. some background knowledge and explain that result. There are various formulations of the problem that have previously been proposed [46, 47, 49, 50, 54]. In general, the idea is to find meaningful reasons that explain the results of semantic matching, while retaining the original knowledge as much as possible. However, in some cases, the results of semantic matching may present in forms that complicate the formulation of explanations, i.e. non-entailments. This paper addresses those cases of semantic matching.

Our main contribution in this paper represents a method for explaining non-entailments as a result of semantic matching for the description logic $\mathcal{EL}_\bot$. We focus on performing abduction between complex concept definitions with the goal of finding *direct relations* between them. In general, the problem of generating explanations for non-entailments is brought down to a search for relations between concepts and/or roles/role restrictions in inclusions and equivalence axioms. Our methodology does not search for relating concepts within a background knowledge, but can be easily extended to do so. The abduction is performed on complex expressions, which are first transformed into corresponding graphical representations. Next, structure preserving maps are produced between the graphical representations of concepts, which provide the necessary relations between those concepts that could be abducted. Hence cases of semantic matching between concepts can be essentially brought down to concept inclusions and equivalence and the concepts can be inputted as complex expressions; we picked this scenario to better illustrate how our method computes abductive solutions between complex concept definitions. We describe our semantic matching scenario and the types of axioms we focus on to explain when they are not entailed.

To solve abduction problems, some methods [16, 19, 28, 37] impose constraints on the set of concepts and/or role restrictions that can be included in explanations. These constraints usually allow for abduction of concepts only, and omit role restrictions. In addition, they limit the types of complex expressions that can be included in explanations. The challenge here is to minimize those constraints and allow for all forms of complex expressions to be included in explanations. To this end, we present our approach using subtree isomorphisms and illustrate how abductive solutions, that contain both concepts and role restrictions, can be constructed. We present a semantic model for the representation of complex definitions consisting of concepts and roles, which is constrained only by the signature of knowledge bases, and formalize our approach. Finally, we present an implementation of our method and illustrate the results for a working example, as well as present experimental results from a synthetically constructed benchmark and on real-world ontology datasets.

The remaining of the paper is structured as follows: Section 2 presents related work on abduction and methods for generating explanations of non-entailments in DLs, how explanation techniques are used for semantic matching, and limitations and challenges in the areas, Section 3 overviews the Description Logic $\mathcal{EL}_\bot$ and semantic matching in DLs, as well as related notions, in Section 4 we define the problem of explaining non-entailments obtained as a result of semantic matching, in Section 5 we present our methodology, Section 6 contains the implementation of our method presented on a working example, as well as results obtained from an experimental setup, Section 7 contains a brief comparison of our method to existing ones and finally, Section 8 contains concluding remarks.

2. Related Work

The problem of explaining results of semantic matching has been treated in various different ways in the past. They can be outlined by two main categories: explaining entailments, and explaining non-entailments. An entailment is an axiom that logically follows from a knowledge-base, e.g. an ontology or a knowledge graph. They are often referred to as logical inferences. One established method for explaining entailments are justifications [10]. A justification is a subset consisting of the minimal number of axioms from an ontology such that the entailment holds. Recently, proofs, in the form of directed hypergraphs, have been utilized to explain inferences for a defined logic $\mathcal{L}$, which could be either first-order logic or some description logic [11–14]. These methods provide excellent results when explaining entailments and, when a result of semantic matching presents as such, they can be applied directly. However, these methods fall short when explaining non-entailments, i.e. axioms that do not logically follow from a knowledge base. In this case, a standard approach is abduction [27, 29, 38, 41, 47, 49, 50, 54].

An abduction problem consists of a background knowledge and an observation that is not entailed. A solution, referred to as a hypothesis, is a set of axioms that when added to the background knowledge the observation becomes entailed. Since the hypothesis provides reasons for why the observation was not entailed in the first place, it is treated as an explanation for the non-entailment. Depending on the background knowledge and the observation, we can perform concept abduction [38, 39], where the hypothesis consists of atomic concepts restricted by the background knowledge, ABox abduction [27, 29–37], where the background knowledge includes assertions and the observation is a query to the ABox (the hypothesis contains assertive knowledge), TBox abduction [15–17], specific for explaining observations in the form of general concept inclusion axioms, and knowledge base abduction [28, 29], where the hypothesis consists of terminological and assertive axioms constructed from the background knowledge.

We can also see the utilization of abduction for semantic matching in [45, 47, 54], where a search is propagated for the assumptions needed for a supply to meet a demand. The focus is mainly on the abduction of concepts that are added or contracted from definitions of matching concepts, in order to achieve a certain degree of semantic similarity between them. The added concepts represent hypotheses that explain the negative outcome of the semantic match. However, this contraction or extension may change the original concept definitions and the explanations may not present as meaningful. To tackle this, specific constraints are introduced in the search for explanations. Our interest lies in preserving the definitions of matching concepts by finding direct relations between them, while putting minimal constraints on the process of abduction.

The problem of evaluating concept compatibility in semantic matching can be brought down to a search for a conjunction (or an extension of a conjunction) for given matching concept definitions. Ultimately, this search can be transformed into the subgraph isomorphism problem [51, 52]. We present how the subgraph isomorphism problem can be utilized to explain non-entailments of semantic matching and a method that computes subtree isomorphisms to generate explanations for non-entailments of semantic matching.

Similarly to [45, 47, 54], abduction is used to generate explanations. However, our method focuses on obtaining direct relations between matching concepts, i.e. general concept inclusions (GCIs), instead of concepts alone, to explain non-entailments of semantic matching. This approach preserves the structure of the definitions of matching concepts, thus filtering explanations that do not provide critical information for the outcome of semantic matching.

While graphs and homomorphisms have been used previously to solve abduction problems and to explain general concept inclusion non-entailments [15], which our method also does, we do not search for relating concepts in TBoxes, but generates explanations of non-entailments that contain direct connections between complex concept definitions, which are obtained as a result of our semantic matching scenario.

In addition, we use subtree isomorphisms instead of homomorphisms between graphical representations of concept definitions to allow for abduction of not only concepts, but also role restrictions, which represents our major contribution in this paper.

Previously, in our work [41] we presented an approach for generating explanations for non-entailments in DLs using subtree isomorphisms. The approach is focused on explaining general concept inclusion non-entailments. Since the method in [41] finds direct relations between concepts in a non-entailment, it fits the use-case of semantic matching. In this paper we formalize our previous work and show how it can be used to generate explanations for results of semantic matching that are not entailed. We extend the definitions for subtrees and show in detail how

subtree isomorphisms capture concept inclusions in semantic matching. In addition, we fully implement and test our method.

3. Preliminaries

In this section, we briefly summarize the description logic $\mathcal{EL}_\bot$, which is an extension of the description logic $\mathcal{EL}$ that also allows concept disjointness, along with other fundamental concepts and notations used throughout this paper.

3.1. $\mathcal{EL}_\bot$ Description Logic

We start by defining countably infinite disjoint sets, $\mathcal{N}_\mathcal{C}$ whose members are called *atomic concepts*, and $\mathcal{N}_\mathcal{R}$ whose members are called *roles*. By combining these we form a *signature*, $\Sigma := \langle \mathcal{N}_\mathcal{C}, \mathcal{N}_\mathcal{R} \rangle$. We denote atomic concepts by A_i, roles by r_i, and complex concepts C_i and/or D_i for $i \in \mathbb{N}^0$. In the case where a distinct concept and/or role occurs, the indexing is omitted and we write A for atomic concept, r for role, and C and/or D for a complex concept.

$\mathcal{EL}_\bot$ *concepts* (a.k.a. *complex concepts*) are inductively defined by the syntax:

$$C, D := \top \mid \bot \mid A \mid C \sqcap D \mid \exists r.C,$$

where $A \in \mathcal{N}_\mathcal{C}$, $r \in \mathcal{N}_\mathcal{R}$.

For concise representation, we denote a conjunction of $\mathcal{EL}_\bot$ concepts by $C_0 \sqcap C_1 \sqcap \cdots \sqcap C_n = \prod_{i=0}^{n} C_i$.

The *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a non-empty set called the *domain* of $\mathcal{I}$ and a function $\cdot^\mathcal{I}$. The function $\cdot^\mathcal{I}$, defined on the sets of atomic concepts $\mathcal{N}_\mathcal{C}$ and roles $\mathcal{N}_\mathcal{R}$, associates with each concept name $A \in \mathcal{N}_\mathcal{C}, A^\mathcal{I} \subseteq \Delta^\mathcal{I}$, and with each role name $r \in \mathcal{N}_\mathcal{R}, r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$. We can extend the *interpretation function* $\cdot^\mathcal{I}$ to complex concepts, by inductively defining:

- $\top^\mathcal{I} := \Delta^\mathcal{I}$, $\bot^\mathcal{I} := \emptyset$,
- $(C \sqcap D)^\mathcal{I} := C^\mathcal{I} \cap D^\mathcal{I}$,
- $(\exists r.C)^\mathcal{I} := \{a \in \Delta^\mathcal{I} \mid \exists b \in \Delta^\mathcal{I}, (a,b) \in r^\mathcal{I} \wedge b \in C^\mathcal{I}\}$.

A *TBox* $\mathcal{T}$ represents terminological knowledge and is a finite set of General Concept Inclusions (GCIs, a.k.a. axioms) of the forms $C \sqsubseteq D$ - we say "*C is subsumed by D*" with the meaning "*C is included in D*" or "*D includes C*" and $C \sqsupseteq D$ - we say "*C subsumes D*" with the meaning "*D is included in C*" or "*C includes D*".

We denote axioms by α and add indexing when there are multiple axioms, α_i for $i \in \mathbb{N}^0$. The interpretation $\mathcal{I}$ is a model of $C \sqsubseteq D$ if $C^\mathcal{I} \subseteq D^\mathcal{I}$ and it is written as $\mathcal{I} \models C^\mathcal{I} \subseteq D^\mathcal{I}$. Similarly, $\mathcal{I}$ is a model of $C \sqsupseteq D$ if $C^\mathcal{I} \supseteq D^\mathcal{I}$, written as $\mathcal{I} \models C^\mathcal{I} \supseteq D^\mathcal{I}$. $\mathcal{I}$ is a model of $C \equiv D$ if $\mathcal{I}$ is a model of both $C \sqsubseteq D$ and $C \sqsupseteq D$. If the interpretation $\mathcal{I}$ is a model of every axiom in a TBox $\mathcal{T}$, then it is a model of $\mathcal{T}$. A TBox $\mathcal{T}$ is consistent if it has a model.

Definition 1. *A TBox $\mathcal{T}$ entails an axiom α if and only if all models of $\mathcal{T}$ satisfy α. In this case, we write $\mathcal{T} \models \alpha$.*

Definition 2. *A TBox $\mathcal{T}$ does not entail an axiom α if there exists a model of $\mathcal{T}$ that does not satisfy α. In this case, we write $\mathcal{T} \not\models \alpha$.*

When an axiom α is entailed by a TBox $\mathcal{T}$, $\mathcal{T} \models \alpha$, we refer to it as an *entailment* and say that α is entailed or α is a logical inference (consequence) of $\mathcal{T}$. When α is not entailed by $\mathcal{T}$, $\mathcal{T} \not\models \alpha$, we call it a *non-entailment* and say that α is not entailed or α is not a logical inference (consequence) of $\mathcal{T}$. The subsumption problem for the $\mathcal{EL}$ family of description logics is decidable in *polynomial time* [43].

3.2. Rooted Trees

We introduce some fundamental concepts and notations from graph theory, that are used throughout this paper. A directed rooted tree is a tuple $T = \langle \mathcal{V}, \mathcal{E}, \rho \rangle$, where:

- $\mathcal{V}$ is a set of nodes,
- $\mathcal{E}$ is a set of edges, which are ordered pairs of distinct nodes $\mathcal{E} = \{\langle x, y \rangle \mid \langle x, y \rangle \in \mathcal{V}^2 \text{ and } x \neq y\}$, which are assigned a natural orientation away from the root,
- ρ is the root node, $\rho \in \mathcal{V}$,
- T is connected, i.e. each node can be reached when traversing the tree from its root, for all $v_i \in T$, $\exists \pi(v_0, v_i)$,
- T is acyclic, i.e. it does not contain any cycles, for all $v_i, v_j \in T$, if $\exists \pi(v_i, v_j)$, then $v_i \neq v_j$.

We denote nodes of trees by $v_i, w_i, u_i, x_i, y_i, z_i$ for $i \in \mathbb{N}^0$ and edges of trees e_k for $k \in \mathbb{N}$. When referring to a distinct node or edge we omit the indexing and write v, w, u, x, y, z for nodes, and e for edges. We use the notation $\langle v_i, v_j \rangle$ to represent an edge e_k. In the edge $e_k = \langle v_i, v_j \rangle$, directed from v_i to v_j, the nodes v_i and v_j are called the *endpoints* of the edge, v_i is the *head* of the edge and v_j is the *tail* of the edge. Edges with the same tail nodes are not allowed.

Let $T = \langle \mathcal{V}, \mathcal{E}, v_0 \rangle$ be a rooted tree. A path π in T is a sequence of nodes $v_1, v_2, \ldots, v_i, \ldots, v_n$ in $\mathcal{V}$ and a sequence of edges $e_1, e_2, \ldots, e_k, \ldots, e_{n-1}$ in $\mathcal{E}$, such that π begins with v_1 and ends with v_n and for each subsequent edges e_k and e_{k+1}, the tail node of e_k is the head node of e_{k+1}. All nodes and all edges in π are distinct. We denote by $\pi(v_1, v_n) = \langle v_1, \ldots, v_n \rangle$ for $v_1, \ldots, v_n \in \mathcal{V}$ the sequence of nodes for the path starting in v_1 and ending in v_n, and denote by $\pi_e(v_1, v_n) = \langle e_1, \ldots, e_{n-1} \rangle$ for $e_1, \ldots, e_{n-1} \in \mathcal{E}$ the sequence of edges for the path starting in v_1 and ending in v_n. $\pi(v_1, v_n)$ and $\pi_e(v_1, v_n)$ are different representations that refer to the same path. In a rooted tree there is a unique path between any two nodes. The length of a path is the number of vertices in a path, and is denoted by $|\pi(v_1, v_n)|$. If v_i lies on the distinct path from the root to v_n, then v_i is called an an *ancestor* of v_n and v_n is a *descendant* of v_i.

The *depth*, $d(v_i)$, of a node v_i is the number of edges in the path $\pi_e(v_0, v_i)$ and is $d(v_i) = |\pi_e(v_0, v_i)| = |\pi(v_0, v_i)| - 1$. The *children* of a node v_i are defined as $N(v_i) := \{v_j \mid v_j \in \mathcal{V} \text{ and } \langle v_i, v_j \rangle \in \mathcal{E}\}$. The *degree* of a node $v_i \in \mathcal{V}$ is $\delta(v_i) := |N(v_i)|$. We refer to a path as *complete* if it starts at the root node and ends in a leaf node. The set of all complete paths Π^0 in T is $\Pi^0 := \{\pi(v_0, v_i) \mid v_0, v_i \in \mathcal{V}, i \neq 0 \text{ and } \delta(v_i) = 0\}$. When we say $\pi(v_0, v_i) \in \Pi^0$ we also imply $\pi_e(v_0, v_i) \in \Pi^0$ and vice versa.

Nodes and edges of rooted trees can contain labels. A labeling is simply a map $f : A \mapsto B$, such that for every element $b \in B$ there is at least one element $a \in A$, such that $b = f(a)$. We can define a node-edge-labeled rooted tree as follows:

Definition 3. *A rooted tree $T = \langle \mathcal{V}, \mathcal{E}, v_0, \lambda_\mathcal{V}, \lambda_\mathcal{E} \rangle$ is called node-edge-labeled (or labeled) where $\lambda_\mathcal{V}$ is a labeling of the nodes, $\lambda_\mathcal{V} : \mathcal{V} \mapsto L_\mathcal{V}$, which maps all nodes to labels in a set of node labels $L_\mathcal{V}$, and $\lambda_\mathcal{E}$ is a labeling $\lambda_\mathcal{E} : \mathcal{E} \mapsto L_\mathcal{E}$, which maps all edges to labels in a set of edge labels $L_\mathcal{E}$. A rooted tree is called only node-labeled if it only contains a labeling of the nodes. Subsequently, a rooted tree is called only edge-labeled if it only contains a labeling of the edges.*

3.3. Semantic Matching

Semantic matching is a technique used to identify semantically related concepts [18, 19, 25, 26, 44]. Introduced in [18], semantic matching determines whether two concepts are semantically similar (or compatible) w.r.t. some background knowledge, if their intersection is satisfiable. If the intersection of two concepts is not satisfiable w.r.t. some background knowledge, then they are not semantically similar, i.e. the concepts are incompatible. We refer to the former as *positive intersection matches* and the latter as *negative intersection matches*.

Definition 4. *Let $\mathcal{T}$ be an $\mathcal{EL}_\bot$ TBox and C and D concepts defined w.r.t. $\Sigma_\mathcal{T} = \langle N_\mathcal{C}, N_\mathcal{R} \rangle$. Positive and negative intersection matches, respectively, occur when:*

$$\mathcal{T} \not\models C \sqcap D \equiv \bot, \tag{1}$$

$$\mathcal{T} \models C \sqcap D \equiv \bot. \tag{2}$$

A positive result of an intersection match (or a positive intersection match) occurs if two concepts C and D are semantically similar. Consequently, a negative result of an intersection match (or a negative intersection match), occurs if two concepts C and D are not semantically similar. Since it is not particularly useful to merely determine if two concepts are semantically similar or not, intersection matches are extended to: *exact*, *plugin*, and *subsume*, as seen in [18, 44]. It would be more useful to know whether they are, or could be, in a higher matching degree. The exact, plugin, and subsume match levels tell us the extent to which the concepts are semantically similar, which is more useful in decision-making scenarios.

Positive and negative semantic matches of higher degrees are defined as follows:

Definition 5. *Let $\mathcal{T}$ be an $\mathcal{EL}_\bot$ TBox and C and D concepts defined w.r.t. $\Sigma_\mathcal{T} = \langle N_\mathcal{C}, N_\mathcal{R} \rangle$. Positive semantic matches of higher degrees occur when:*

$$\textbf{\textit{Exact}} : \mathcal{T} \models C \equiv D, \tag{3}$$

$$\textbf{\textit{PlugIn}} : \mathcal{T} \models C \sqsubseteq D, \tag{4}$$

$$\textbf{\textit{Subsume}} : \mathcal{T} \models C \sqsupseteq D. \tag{5}$$

and negative semantic matches of higher degrees occur when:

$$\textbf{\textit{Exact}} : \mathcal{T} \not\models C \equiv D, \tag{6}$$

$$\textbf{\textit{PlugIn}} : \mathcal{T} \not\models C \sqsubseteq D, \tag{7}$$

$$\textbf{\textit{Subsume}} : \mathcal{T} \not\models C \sqsupseteq D. \tag{8}$$

Let C and D be two concepts defined w.r.t. the signature of some background knowledge. We denote by $\text{match}_\sqcap(C, D)$ the intersection match between the concepts $(C \sqcap D \sqsubseteq \bot)$, by $\text{match}_\equiv(C, D)$ the exact match between the concepts $(C \equiv D)$, by $\text{match}_\sqsubseteq(C, D)$ the plugin match between the concepts $(C \sqsubseteq D)$, and by $\text{match}_\sqsupseteq(C, D)$ the subsume match between the concepts $(C \sqsupseteq D)$. When we write $\text{match}_{\underline{\square}}(C, D)$ we mean any one of the exact, plugin, or subsume match. The symbol $\underline{\square}$ is a placeholder for $(\equiv, \sqsubseteq, \text{or } \sqsupseteq)$.

4. Problem Formulation

There is a clear distinction between the types of semantic matches and results they provide, as well as the way the results present themselves w.r.t. some background knowledge. Intersection matches provide only information whether the concepts in the semantic match are compatible or not. For example, if two concepts are not compatible, the (negative) result of the intersection match between them presents as an entailment and to explain it we can invoke a justification, which will provide the reasons why the concepts are not compatible. However, if two concepts are, in fact, compatible, it presents as a non-entailment. To explain this (positive) result of their intersection match, we refer to exact, plugin, or subsume matches. To put it simply, the information that the positive intersection match gave can be further used to identify the degree to which the matching concepts *could be* semantically similar. If we explain the extent of their similarity, we also inherently explain why they are compatible.

Let $\mathcal{T}$ be an $\mathcal{EL}_\bot$ TBox and $\text{match}_{\underline{\square}}(C, D)$ a semantic match of higher degree (exact, plugin, or subsume), such that the outcome of the semantic match is negative, $\mathcal{T} \not\models \text{match}_{\underline{\square}}(C, D)$. A classical approach to explain this non-entailment is *abduction*, i.e. finding "missing" knowledge such that when added to $\mathcal{T}$ the result of the semantic match becomes positive. If the matching concepts should, in fact, be in a positive exact, plugin, or subsume semantic relation, i.e. if $\text{match}_{\underline{\square}}(C, D)$ should logically follow from $\mathcal{T}$, then abduction allows us to find reasons why it was not entailed and fix the non-entailment. The abduction problem we are interested in for semantic matching is defined as follows:

Definition 6. *Let $\mathcal{T}$ be an $\mathcal{EL}_\bot$ TBox and $match_{\underline{\sqsubset}}(C,D)$ an exact ($C \equiv D$), plugin ($C \sqsubseteq D$), or subsume ($C \sqsupseteq D$) semantic match between concepts C and D. An abduction problem is a tuple $\langle \mathcal{T}, match_{\underline{\sqsubset}}(C,D)\rangle$, where $\mathcal{T}$ is the background knowledge, $\mathcal{T} \not\models C \sqcap D \equiv \bot$ and $\mathcal{T} \not\models match_{\underline{\sqsubset}}(C,D)$, i.e. the result of the semantic match is negative (non-entailment). A solution to the abduction problem is a hypothesis $\mathcal{H}$ of the form:*

$$\mathcal{H} = \{\alpha \mid \mathcal{T} \not\models \alpha\},$$

such that $\mathcal{T} \cup \mathcal{H} \not\models C \sqcap D \equiv \bot$ and $\mathcal{T} \cup \mathcal{H} \models match_{\underline{\sqsubset}}(C,D)$.

5. Explaining Non-entailments of Semantic Matching

To explain non-entailments of semantic matching we opt to search for "missing" connections between concept descriptions introduced in a negative match. The case of intersection matching is used purely to identify whether the background knowledge can, in fact, provide a consistent model of a semantic relation between matching concepts. For the negative outcomes of higher degree matches, we need to construct hypotheses that will contain the axioms such that when added to the background knowledge the outcome of the match becomes positive. To do this, we need to take into account the following:

Point 1. Hypotheses should contain axioms that preserve the structure of concept descriptions (definitions) introduced in matches,
Point 2. Hypotheses should take into account the structure of concept descriptions that cannot be preserved,
Point 3. Hypotheses should not unnecessarily omit parts of concept descriptions that preserve the structure.

The three points refer to the overall idea of performing abduction between matching concepts. **Point 1** addresses the general settings of concept inclusions and equivalences. To give an example, consider some concepts defined as $C \equiv A_0 \sqcap \exists r.A_1$ and $D \equiv B_0 \sqcap \exists r.B_1$ w.r.t. some terminological knowledge $\mathcal{T}$. In order for $\mathcal{T} \models C \sqsubseteq D$, the background knowledge should contain the information $\mathcal{T} \models A_0 \sqsubseteq B_0$ and $\mathcal{T} \models A_1 \sqsubseteq B_1$. If $\mathcal{T} \models A_0 \sqsubseteq B_1$ and $\mathcal{T} \models A_1 \sqsubseteq B_0$, then we would not have the necessary information for $C \sqsubseteq D$ to hold w.r.t. $\mathcal{T}$. We can see that the required knowledge is structured, i.e. the top level concept in one definition is subsumed by the top level concept in the other, and the concept restricted by the role in the first definition is subsumed by the concept restricted with the same role in the second definition. This structure must be preserved in order to correctly compute abductive solutions (hypotheses).

However, in some cases, this structure cannot be preserved. Take for example the concepts $C \equiv A_0 \sqcap \exists r.A_1$ and $D \equiv B_0 \sqcap \exists s.B_1$. We can see that A_0 can be related to B_0, but in C the concept A_1 is restricted by the role r and in D the concept B_1 is restricted by the role s. If we have that $\mathcal{T} \models A_1 \sqsubseteq B_1$, we would still not have the critical knowledge in order for $C \sqsubseteq D$ to be entailed by $\mathcal{T}$. This is because the concepts are restricted by different roles and **Point 2** refers exactly to cases such as this one. Here, we would need to take into account the entire role restrictions in C and D, in order to obtain the necessary knowledge for the axiom to hold, i.e. $\mathcal{T} \models A_0 \sqsubseteq B_0$ and $\mathcal{T} \models \exists r.A_1 \sqsubseteq \exists s.A_2$.

Finally, **Point 3** addresses the intuition behind constructing abductive solutions (hypotheses). To exemplify this, consider again the concepts $C \equiv A_0 \sqcap \exists r.A_1$ and $D \equiv B_0 \sqcap r.B_1$. In order for $\mathcal{T} \models C \sqsubseteq D$, we need for $\mathcal{T} \models A_0 \sqsubseteq B_0$ and $\mathcal{T} \models A_1 \sqsubseteq B_1$, which $A_0 \sqsubseteq B_0$ and $A_1 \sqsubseteq B_1$ would be our abductions. However, we can also abduct $A_0 \sqcap r.A_1 \sqsubseteq B_0 \sqcap \exists r.B_1$, which is a more complex expression. If $\mathcal{T} \models A_0 \sqcap r.A_1 \sqsubseteq B_0 \sqcap \exists r.B_1$, then $\mathcal{T} \models C \sqsubseteq D$, but in this case we would obtain a more complex abduction, rather than two simpler ones that are easier to understand, and most importantly, lead to the same outcome.

The idea of preserving the structure in **Point 1** refers to avoiding abduction of unrelated concepts. We present the notion of an $\mathcal{EL}$ *description tree*, as originally defined in [40] and revisited in [15], which is a graphical representation of $\mathcal{EL}$ concepts.

Definition 7. *Let $\mathcal{T}$ be an $\mathcal{EL}_\bot$ TBox with signature $\Sigma = \langle N_C, N_R\rangle$. An $\mathcal{EL}$ description tree is a labeled directed rooted tree, $T = \langle \mathcal{V}, \mathcal{E}, v_0, \lambda_\mathcal{V}, \lambda_\mathcal{E}\rangle$, where:*

– *$\mathcal{V}$ is a finite set of nodes,*
– *$\mathcal{E}$ is a finite set of edges,*
– *v_0 is the root node, s.t. $v_0 \in \mathcal{V}$,*
– *$\lambda_{\mathcal{V}} : \mathcal{V} \mapsto \mathcal{N}_{\mathcal{C}}$, such that for any $v \in \mathcal{V}$, $\lambda_{\mathcal{V}}(v) \subseteq \mathcal{N}_{\mathcal{C}}$, and*
– *$\lambda_{\mathcal{E}} : \mathcal{E} \mapsto \mathcal{N}_{\mathcal{R}}$, such that for any $e \in \mathcal{E}$, $\lambda_{\mathcal{E}}(e) \in \mathcal{N}_{\mathcal{R}}$.*

The empty label represents the top concept ($\top$). The bottom concept ($\bot$) is not allowed in description trees.

We can recursively define a concept C which is represented by a description tree. Let $T_C = \langle \mathcal{V}_C, \mathcal{E}_C, v_0, \lambda_{\mathcal{V}_C}, \lambda_{\mathcal{E}_C} \rangle$ be the description tree for the concept definition C. If $v_1, v_2, \ldots, v_n$ are the children of v_0 and we denote by $T_C(v_1), T_C(v_2), \ldots, T_C(v_n)$ the pairwise disjoint subtrees of T_C rooted in $v_1, v_2, \ldots, v_n$, then we can define the concept C as:

$$C = C_{T_C(v_0)}, \tag{9}$$
$$C_{T_C(v_i)} = \prod_{v_i \in \mathcal{V}} \lambda_{\mathcal{V}_C}(v_i) \sqcap \prod_{\langle v_i, v_j \rangle \in \mathcal{E}_C} \exists \lambda_{\mathcal{E}_C}(\langle v_i, v_j \rangle).C_{T_C(v_j)},$$

where $C_{T(v_0)}$ is the concept represented by the tree rooted in v_0, and $C_{T(v_i)}$ are the concepts represented by the trees rooted in v_i.

Likewise, if we have a concept definition $C \equiv \prod_{i=0}^{n} A_i \sqcap \exists r_1.C_1 \sqcap \cdots \sqcap \exists r_n.C_n$, we can define the description tree $T_C = \langle \mathcal{V}_C, \mathcal{E}_C, v_0, \lambda_{\mathcal{V}_C}, \lambda_{\mathcal{E}_C} \rangle$ of C inductively, based on the pairwise disjoint description trees $T_{C_1}, T_{C_2}, \ldots, T_{C_n}$ for concepts $C_1, C_2, \ldots, C_n$ in the definition of C. If we denote by $T_i = \langle \mathcal{V}_i, \mathcal{E}_i, v_i, \lambda_{\mathcal{V}_i}, \lambda_{\mathcal{E}_i} \rangle$ the description trees for concepts C_i, then:

$$\mathcal{V}_C = \{v_0\} \bigcup_{i=1}^{n} \mathcal{V}_i,\ \mathcal{E}_C = \{\langle v_0, v_i \rangle \mid 1 \leqslant i \leqslant n\} \bigcup_{i=1}^{n} \mathcal{E}_i,\ \lambda_{\mathcal{V}_C}(v_0) = \{A_0, A_1, \ldots, A_n\}, \tag{10}$$
$$\lambda_{\mathcal{V}_C}(v_i) = \lambda_{\mathcal{V}_i}(v_i), \text{for any } v_i \in \mathcal{V}_i,$$
$$\lambda_{\mathcal{E}_C}(\langle v_i, v_j \rangle) = \lambda_{\mathcal{E}_i}(\langle v_i, v_j \rangle), \text{for any } \langle v_i, v_j \rangle \in \mathcal{E}_i.$$

Representing concept definitions as description trees gives the potential to relate concepts w.r.t. a background knowledge. For example, homomorphisms between description trees capture subsumption between $\mathcal{EL}$ concepts [15, 40]. This is achieved by generating a mapping between the vertex sets of description trees which preserves the structure between edges and labels. Consequently, this structure preserving map translates in the form of a concept inclusion, w.r.t. a background knowledge, between the concepts represented by those description trees. We focus on description tree isomorphisms, as a means of capturing subsumption and equivalence relations between concept definitions in semantic matching.

Definition 8. *Let $T_C = \langle \mathcal{V}_C, \mathcal{E}_C, v_0, \lambda_{\mathcal{V}_C}, \lambda_{\mathcal{E}_C} \rangle$ and $T_D = \langle \mathcal{V}_D, \mathcal{E}_D, w_0, \lambda_{\mathcal{V}_D}, \lambda_{\mathcal{E}_D} \rangle$ be two description trees. A weak isomorphism from T_C to T_D is a bijective mapping $\phi : T_C \mapsto T_D$, such that:*

1. *$\phi(v_0) = w_0$,*
2. *$\langle v_i, v_j \rangle \in \mathcal{E}_C \Leftrightarrow \langle \phi(v_i), \phi(v_j) \rangle \in \mathcal{E}_D$ and $\lambda_{\mathcal{E}_C}(\langle v_i, v_j \rangle) = \lambda_{\mathcal{E}_D}(\langle \phi(v_i), \phi(v_j) \rangle)$.*

We write $T_C \cong T_D$ if two trees are weakly isomorphic.

We propagate the notation from [15], and distinguish description tree isomorphisms in two main ways: *weak isomorphisms*, i.e. isomorphisms that satisfy the conditions in Definition 8, and *$\mathcal{T}$-isomorphisms*, which are weak isomorphisms that contain a semantic layer. Further, we categorize $\mathcal{T}$-isomorphisms into three types:

– Type **exact** $\mathcal{T}$-isomorphism, denoted by $\mathcal{T}_{\equiv}$-isomorphism,
– Type **plugin** $\mathcal{T}$-isomorphism, denoted by $\mathcal{T}_{\sqsubseteq}$-isomorphism,
– Type **subsume** $\mathcal{T}$-isomorphism, denoted by $\mathcal{T}_{\sqsupseteq}$-isomorphism.

Definition 9. *Let $\mathcal{T}$ be an $\mathcal{EL}_\bot$ TBox and $T_C = \langle \mathcal{V}_C, \mathcal{E}_C, v_0, \lambda_{\mathcal{V}_C}, \lambda_{\mathcal{E}_C} \rangle$ and $T_D = \langle \mathcal{V}_D, \mathcal{E}_D, w_0, \lambda_{\mathcal{V}_D}, \lambda_{\mathcal{E}_D} \rangle$ be two description trees. A weak isomorphism $\phi : T_C \mapsto T_D$ becomes:*

1. ***Exact*** *($\mathcal{T}_\equiv$-isomorphism), when $\forall v \in \mathcal{V}_C$, and $w \in \mathcal{V}_D$ s.t. $\phi(v) = w$, $\mathcal{T} \models \bigsqcap \lambda_{\mathcal{V}_D}(w) \equiv \bigsqcap \lambda_{\mathcal{V}_C}(v)$,*
2. ***Plugin*** *($\mathcal{T}_\sqsubseteq$-isomorphism), when $\forall v \in \mathcal{V}_C$, and $w \in \mathcal{V}_D$ s.t. $\phi(v) = w$, $\mathcal{T} \models \bigsqcap \lambda_{\mathcal{V}_D}(w) \sqsubseteq \bigsqcap \lambda_{\mathcal{V}_C}(v)$,*
3. ***Subsume*** *($\mathcal{T}_\sqsupseteq$-isomorphism), when $\forall v \in \mathcal{V}_C$, and $w \in \mathcal{V}_D$ s.t. $\phi(v) = w$, $\mathcal{T} \models \bigsqcap \lambda_{\mathcal{V}_D}(w) \sqsupseteq \bigsqcap \lambda_{\mathcal{V}_C}(v)$.*

Each type of a $\mathcal{T}$-isomorphism characterizes specific concept relations, i.e. $\mathcal{T}_\equiv$-isomorphism captures equivalence, and $\mathcal{T}_\sqsubseteq$-isomorphism and $\mathcal{T}_\sqsupseteq$-isomorphism capture inclusion. Consequently, the three types of $\mathcal{T}$-isomorphisms capture relations for corresponding types of a match, i.e. $\mathcal{T}_\equiv$-isomorphism captures relations in exact matches, $\mathcal{T}_\sqsubseteq$-isomorphism in plugin, and $\mathcal{T}_\sqsupseteq$-isomorphism in subsume. The semantic layer in Definition 9 extends weak isomorphisms and allows for TBoxes to capture equivalence and subsumption, which is also convenient for capturing the relations in exact, plugin, and subsume matches.

Theorem 1. *Let $\mathcal{T}$ be an $\mathcal{EL}_\bot$ TBox, C and D two concepts defined w.r.t. $\mathcal{T}$, and $T_C = \langle \mathcal{V}_C, \mathcal{E}_C, v_0, \lambda_{\mathcal{V}_C}, \lambda_{\mathcal{E}_C} \rangle$ the description tree of C and $T_D = \langle \mathcal{V}_D, \mathcal{E}_D, w_0, \lambda_{\mathcal{V}_D}, \lambda_{\mathcal{E}_D} \rangle$ description tree of D. If there exists a $\mathcal{T}_\equiv$-isomorphism $\phi : T_C \mapsto T_D$, then $\mathcal{T} \models D \equiv C$.*

Proof. We are examining the case of $\mathcal{T} \models D \equiv C$. The proofs for the cases $\mathcal{T} \models D \sqsubseteq C$ and $\mathcal{T} \models D \sqsupseteq C$ are analogous.

The proof is done by structural induction on the depth of T_C. Assume $\phi : T_C \mapsto T_D$ to be a $\mathcal{T}_\equiv$-isomorphism.

Base case: If $T_C = \langle \{v_0\}, \emptyset, v_0, \lambda_{\mathcal{V}_C}, \lambda_{\mathcal{E}_C} \rangle$ and $T_D = \langle \{w_0\}, \emptyset, w_0, \lambda_{\mathcal{V}_D}, \lambda_{\mathcal{E}_D} \rangle$, we have that $C \equiv \bigsqcap \lambda_{\mathcal{V}_C}(v_0)$ and $D \equiv \bigsqcap \lambda_{\mathcal{V}_D}(w_0)$. Since $\mathcal{T} \models \bigsqcap \lambda_{\mathcal{V}_D}(w_0) \equiv \bigsqcap \lambda_{\mathcal{V}_C}(v_0)$ and $\phi(v_0) = w_0$, it follows that $\mathcal{T} \models D \equiv C$.

Induction: We extend the tree $T_C = \langle \{v_0\} \cup \{v_i \mid 1 \leqslant i \leqslant n\}, \{\langle v_0, v_i \rangle \mid 1 \leqslant i \leqslant n\}, v_0, \lambda_{\mathcal{V}_C}, \lambda_{\mathcal{E}_C} \rangle$, s.t. each v_i is the root of the subtree $T_C(v_i)$ in T_C. Since ϕ is a $\mathcal{T}_\equiv$-isomorphism from T_C to T_D, i.e. a bijective mapping, for each child of v_0 in T_C there is a corresponding child of w_0 in T_D, s.t. $\phi(v_i) = w_i$.

Assuming that ϕ is also a $\mathcal{T}_\equiv$-isomorphism from $T_C(v_i)$ to $T_D(w_i)$, by induction we have that $\mathcal{T} \models D_{T_D(w_i)} \equiv C_{T_C(v_i)}$ and subsequently $\mathcal{T} \models \exists \lambda_{\mathcal{E}_D}(\langle w_0, w_i \rangle).D_{T_D(w_i)} \equiv \exists \lambda_{\mathcal{E}_C}(\langle v_0, v_i \rangle).C_{T_C(v_i)}$ for all children of v_0 in T_C and w_0 in T_D. Because $\mathcal{T} \models \bigsqcap \lambda_{\mathcal{V}_D}(w_0) \equiv \bigsqcap \lambda_{\mathcal{V}_C}(v_0)$ we have that $\mathcal{T} \models D \equiv C$. □

Considering that $\mathcal{T}$-isomorphisms preserve the structure between concept definitions and characterize concept equivalence and subsumption in $\mathcal{EL}_\bot$, they can be used to solve abduction problems as defined in Definition 6. Particularly, the concept equivalence and inclusions from Definition 9 extend weak isomorphisms to certain types of $\mathcal{T}$-isomorphisms. Thus, we can search for weak isomorphisms between description trees, and, by adding a semantic layer in the form of a hypothesis we can extend them to, in fact, $(\mathcal{T} \cup \mathcal{H})$-isomorphisms where needed. The hypothesis $\mathcal{H}$ will contain "missing" knowledge that explains the non-entailment. Dependent on the type of the semantic match we are interested in, we can perform abduction with either ($\equiv$) (to represent exact matches), ($\sqsubseteq$) (to represent a more specific match) or ($\sqsupseteq$) (to represent a less specific match).

Definition 10. *Let $\mathcal{T}$ be an $\mathcal{EL}_\bot$ TBox, C and D two concepts defined w.r.t. $\mathcal{T}$, such that $\mathcal{T} \not\models match_{\underline{\square}}(D, C)$. If there exists a weak isomorphism $\phi : T_C \mapsto T_D$, ϕ can be extended to a $\mathcal{T}_{\underline{\square}}$-isomorphism by constructing a hypothesis $\mathcal{H}$, such that:*

$$\mathcal{H} = \{ \bigsqcap_{w \in \mathcal{V}_D} \lambda_{\mathcal{V}_D}(w) \underline{\square} \bigsqcap_{v \in \mathcal{V}_C} \lambda_{\mathcal{V}_C}(v) \mid \forall v \in \mathcal{V}_C, w \in \mathcal{V}_D \text{ s.t. } \phi(v) = w \} \tag{11}$$

To illustrate how a $\mathcal{T}$-isomorphism captures concept relations and how weak isomorphisms can be extended via a hypothesis to $\mathcal{T}$-isomorphisms, consider the following example.

Example 1. *Let $\mathcal{T}$ be the TBox $\mathcal{T} = \{B_0 \sqsubseteq A_0, B_1 \sqsubseteq A_1, B_2 \sqsubseteq A_2, B_1 \sqsubseteq A_4\}$, and let:*

$$C_1 = A_0 \sqcap \exists r.A_1 \sqcap \exists s.A_2,$$
$$D = B_0 \sqcap \exists r.B_1 \sqcap \exists s.B_2,$$
$$C_2 = A_3 \sqcap \exists r.A_4 \sqcap \exists s.A_5,$$

be three concept descriptions, such that $\mathcal{T} \not\models C_1 \sqcap D \equiv \bot$, $\mathcal{T} \not\models C_2 \sqcap D \equiv \bot$. The description trees of C_1, D, and C_2 are defined as follows:

$$\begin{aligned}
T_{C_1} = \langle \mathcal{V}_{C_1} &= \{v_0, v_1, v_2\}, \mathcal{E}_{C_1} = \{\langle v_0, v_1\rangle, \langle v_0, v_2\rangle\}, v_0,\\
\lambda_{\mathcal{V}_{C_1}} &= \{v_0 \mapsto \{A_0\}, v_1 \mapsto \{A_1\}, v_2 \mapsto \{A_2\}\}, \lambda_{\mathcal{E}_{C_1}} = \{\langle v_0, v_1\rangle \mapsto r, \langle v_0, v_2\rangle \mapsto s\}\rangle,\\
T_D = \langle \mathcal{V}_D &= \{w_0, w_1, w_2\}, \mathcal{E}_D = \{\langle w_0, w_1\rangle, \langle w_0, w_2\rangle\}, w_0,\\
\lambda_{\mathcal{V}_D} &= \{w_0 \mapsto \{B_0\}, w_1 \mapsto \{B_1\}, w_2 \mapsto \{B_2\}\}, \lambda_{\mathcal{E}_D} = \{\langle w_0, w_1\rangle \mapsto r, \langle w_0, w_2\rangle \mapsto s\}\rangle,\\
T_{C_2} = \langle \mathcal{V}_{C_2} &= \{u_0, u_1, u_2\}, \mathcal{E}_{C_2} = \{\langle u_0, u_1\rangle, \langle u_0, u_2\rangle\}, u_0,\\
\lambda_{\mathcal{V}_{C_2}} &= \{u_0 \mapsto \{A_3\}, u_1 \mapsto \{A_4\}, u_2 \mapsto \{A_5\}\}, \lambda_{\mathcal{E}_{C_2}} = \{\langle u_0, u_1\rangle \mapsto r, \langle u_0, u_2\rangle \mapsto s\}\rangle.
\end{aligned}$$

and are illustrated in Figure 1, along with two weak isomorphisms:

1) $\phi_1 : T_{C_1} \mapsto T_D = (v_0 w_0)(v_1 w_1)(v_2 w_2)$ between T_{C_1} and T_D shown in green double sided arrows, and
2) $\phi_2 : T_{C_2} \mapsto T_D = (u_0 w_0)(u_1 w_1)(u_2 w_2)$ between T_{C_2} and T_D shown in blue double sided arrows.

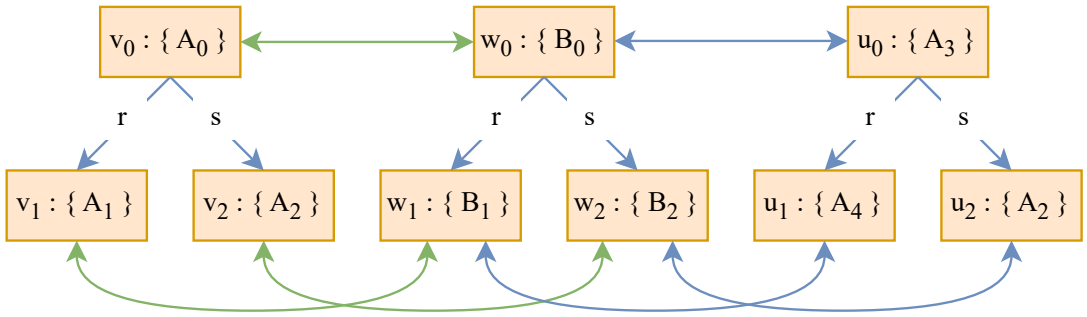

Fig. 1. Example description trees T_{C_1} (left), T_D (middle), and T_{C_2} (right) for concepts C_1, D, and C_2, respectively.

We can notice that ϕ_1 is a $\mathcal{T}_{\sqsubseteq}$-isomorphism, since it is a weak isomorphism between T_{C_1} and T_D, and it also holds that:

- *$\mathcal{T} \models \lambda_{\mathcal{V}_D}(w_0) \sqsubseteq \lambda_{\mathcal{V}_{C_1}}(v_0)$, i.e. $\mathcal{T} \models B_0 \sqsubseteq A_0$,*
- *$\mathcal{T} \models \lambda_{\mathcal{V}_D}(w_1) \sqsubseteq \lambda_{\mathcal{V}_{C_1}}(v_1)$, i.e. $\mathcal{T} \models B_1 \sqsubseteq A_1$, and*
- *$\mathcal{T} \models \lambda_{\mathcal{V}_D}(w_2) \sqsubseteq \lambda_{\mathcal{V}_{C_1}}(v_2)$, i.e. $\mathcal{T} \models B_2 \sqsubseteq A_2$.*

From this, it follows that $\mathcal{T} \models D \sqsubseteq C_1$ and we can see how a $\mathcal{T}$-isomorphism captures concept inclusion.

On the contrary, ϕ_2 is only a weak isomorphism, because the trees T_{C_2} and T_D are structurally isomorphic, but the condition for the semantic layer of a $\mathcal{T}$-isomorphism does not hold. Here, we have:

- *$\mathcal{T} \models \lambda_{\mathcal{V}_D}(w_1) \sqsubseteq \lambda_{\mathcal{V}_{C_2}}(u_1)$, i.e. $\mathcal{T} \models B_1 \sqsubseteq A_4$,*

and

– $\mathcal{T} \not\models \lambda_{\mathcal{V}_D}(w_0) \sqsubseteq \lambda_{\mathcal{V}_{C_2}}(u_0)$, *i.e.* $\mathcal{T} \not\models B_0 \sqsubseteq A_3$, *and*
– $\mathcal{T} \not\models \lambda_{\mathcal{V}_D}(w_2) \sqsubseteq \lambda_{\mathcal{V}_{C_2}}(u_2)$, *i.e.* $\mathcal{T} \not\models B_2 \sqsubseteq A_5$.

Therefore, in this case $\mathcal{T} \not\models D \sqsubseteq C_2$. *However,* $\mathcal{T} \not\models B_0 \sqsubseteq A_3$ *and* $\mathcal{T} \not\models B_2 \sqsubseteq A_5$ *can be considered as "missing" knowledge from* $\mathcal{T}$, *because if they hold then* $\mathcal{T} \models D \sqsubseteq C_2$. *Thus, we can formulate a hypothesis:*

$$\mathcal{H} = \{B_0 \sqsubseteq A_3, B_2 \sqsubseteq A_5\},$$

that extends the weak isomorphism ϕ_2 *to a* $(\mathcal{T} \cup \mathcal{H})_{\sqsubseteq}$*-isomorphism, while preserving the structure between the descriptions trees, and we have that* $\mathcal{T} \cup \mathcal{H} \not\models C_2 \sqcap D \equiv \bot$ *and* $\mathcal{T} \cup \mathcal{H} \models D \sqsubseteq C_2$.

If relations between concepts are missing from some background knowledge $\mathcal{T}$, a weak isomorphism between the graphical representations of those concepts will point to which of those relations are missing. Thus, a semantic layer can be created on top using the weak isomorphism. This semantic layer is exactly the hypothesis that will explain the non-entailment. In other words the semantic layer will extend the weak isomorphism to a $\mathcal{T}$-isomorphism. In addition, weak isomorphisms and $\mathcal{T}$-isomorphisms preserve the structure between description trees of concepts and do not allow for unrelated concepts to be in hypotheses. With this **Point 1** is addressed. However, we need to extend the approach to be able to also abduct role restrictions.

Point 2 addresses the abduction of role restrictions. The key reason why a non-entailment exists may be a role restriction on a certain concept (or complex expression), that depicts a part of the structure of concept descriptions that cannot be preserved. Roles are presented as labels of edges in description trees. If the labels of edges between description trees are different, i.e. not preserved, then by Definition 8 those trees cannot be weakly isomorphic. This also means that the concepts in the descriptions cannot be related and a $\mathcal{T}$-isomorphism (or a $(\mathcal{T} \cup \mathcal{H})$-isomorphism) cannot be obtained. However, if we contract entire parts of descriptions trees, and, consequently of concept definitions, that cannot be preserved and introduce them in hypotheses, then a $(\mathcal{T} \cup \mathcal{H})$-isomorphism could still be reached. By taking away those parts in the description trees, we essentially take away whole nested role restrictions in the original concept definitions. The parts of trees that cannot be preserved will provide the "missing" role restrictions that need to be included in the abductive solution in order for a non-entailment to become entailed. Therefore, abduction of role restrictions is crucial when concepts' definitions cannot be preserved.

To allow for abduction of role restrictions we introduce subtree isomorphisms between $\rho_{\sqsubseteq}$-subtrees.

Definition 11. *Let* $T = \langle \mathcal{V}, \mathcal{E}, v_0, \lambda_{\mathcal{V}}, \lambda_{\mathcal{E}} \rangle$ *be a description tree. Let* $\mathcal{S} \subseteq \mathcal{V}$ *be any subset of vertices of* T. *Then, the induced subtree* $T[\mathcal{S}] = \langle \mathcal{S}, \mathcal{E}_{[\mathcal{S}]}, \rho_{\mathcal{S}}, \lambda_{[\mathcal{S}]}, \lambda_{\mathcal{E}_{[\mathcal{S}]}} \rangle$ *is the subtree of* T *whose node set is* $\mathcal{S}$, *and:*

– $T[\mathcal{S}]$ *is rooted in* $\rho_{\mathcal{S}}$,
– *for any two nodes* $v, u \in \mathcal{S}$, $\langle v, u \rangle \in \mathcal{E}_{[\mathcal{S}]}$ *if* $\langle v, u \rangle \in \mathcal{E}$,
– *for all* $v \in \mathcal{S}$, $\lambda_{[\mathcal{S}]}(v) = \lambda_{\mathcal{V}}(v)$,
– *for all* $e \in \mathcal{E}_{[\mathcal{S}]}$, $\lambda_{\mathcal{E}_{[\mathcal{S}]}}(e) = \lambda_{\mathcal{E}}(e)$.

If $T[\mathcal{S}]$ *is an induced subtree of* T, *we write* $T[\mathcal{S}] \subseteq T$. $T[\mathcal{S}]$ *is called a* $\rho_{\sqsubseteq}$*-subtree if* $T[\mathcal{S}]$ *is an induced subtree of* T *and has the same root as* T, $\rho_{\mathcal{S}} = v_0$, *and denote it by* $T[\mathcal{S}] \subseteq_{\rho} T$.

Observation 1. *A* $\rho_{\sqsubseteq}$*-subtree is a description tree itself.*

Proof. This follows from the definition of $\rho_{\sqsubseteq}$-subtrees. We directly prove this.

A $\rho_{\sqsubseteq}$-subtree $T[\mathcal{S}] = \langle \mathcal{S}, \mathcal{E}_{[\mathcal{S}]}, v_0, \lambda_{[\mathcal{S}]}, \lambda_{\mathcal{E}_{[\mathcal{S}]}} \rangle$ of a description tree $T = \langle \mathcal{V}, \mathcal{E}, v_0, \lambda_{\mathcal{V}}, \lambda_{\mathcal{E}} \rangle$ has the same characteristics as a description tree.

– By definition $\mathcal{S} \subseteq \mathcal{V}$. Thus, $T[\mathcal{S}]$ has a finite set of vertices.
– If $\langle v, u \rangle \in \mathcal{E}$ for any two vertices $v, u \in \mathcal{S}$, then it is also true that $\langle v, u \rangle \in \mathcal{E}_{[\mathcal{S}]}$. Thus, $\mathcal{E}_{[\mathcal{S}]} \subseteq \mathcal{E}$ and $T[\mathcal{S}]$ has a finite set of edges.
– By definition of $\rho_{\sqsubseteq}$-subtrees, $T[\mathcal{S}]$ has the same root as T.
– By definition, for all $v \in \mathcal{S}$ we have $\lambda_{[\mathcal{S}]}(v) = \lambda_{\mathcal{V}}(v)$. Since $\lambda_{\mathcal{V}}(v) \subseteq N_C$, we also have $\lambda_{[\mathcal{S}]}(v) \subseteq N_C$.
– By definition, for all $e \in \mathcal{E}_{[\mathcal{S}]}$ we have $\lambda_{\mathcal{E}_{[\mathcal{S}]}}(e) = \lambda_{\mathcal{E}}(e)$. Since $\lambda_{\mathcal{E}}(e) \in N_R$, we also have $\lambda_{\mathcal{E}_{[\mathcal{S}]}}(e) \in N_R$.

Thus, a $\rho_{\subseteq}$-subtree is a description tree itself. □

Observation 2. *If $T = \langle \mathcal{V}, \mathcal{E}, v_0, \lambda_{\mathcal{V}}, \lambda_{\mathcal{E}} \rangle$ is a description tree, then T is a $\rho_{\subseteq}$-subtree of itself, i.e. $T \subseteq_{\rho} T$.*

Proof. The proof directly follows the definition of an induced subtree. We want to show for a description tree $T = \langle \mathcal{V}, \mathcal{E}, v_0, \lambda_{\mathcal{V}}, \lambda_{\mathcal{E}} \rangle$ that $T \subseteq_{\rho} T$. Then, T is the induced subtree of T, by taking all vertices of T as the inducing subset $\mathcal{S}$, $\mathcal{S} = \mathcal{V}$. Since, the root v_0 is the same, $T \subseteq_{\rho} T$. □

Because $\rho_{\subseteq}$-subtrees are description trees, definition 8 refers to weak isomorphisms between $\rho_{\subseteq}$-subtrees as well. In addition, we can extend weak isomorphisms to $\mathcal{T}$-isomorphisms for $\rho_{\subseteq}$-subtrees, thus providing a semantic layer. In order to do this, we need to take into account parts of descriptions trees that do not preserve the structure, which will provide the missing role restrictions in the abductive solutions.

Definition 12. *Let $\mathcal{T}$ be an $\mathcal{EL}_{\bot}$ TBox and $T_1 = \langle \mathcal{V}_1, \mathcal{E}_1, v_0, \lambda_{\mathcal{V}_1}, \lambda_{\mathcal{E}_1} \rangle$ and $T_2 = \langle \mathcal{V}_2, \mathcal{E}_2, w_0, \lambda_{\mathcal{V}_2}, \lambda_{\mathcal{E}_2} \rangle$ be two description trees. A weak isomorphism $\phi : T_1[\mathcal{S}_1] \mapsto T_2[\mathcal{S}_2]$ for $T_1[\mathcal{S}_1] \subseteq_{\rho} T_1$ and $T_2[\mathcal{S}_2] \subseteq_{\rho} T_2$ becomes:*

1. ***Exact*** *($\mathcal{T}_{\equiv}$)-isomorphism, when $\forall v \in \mathcal{S}_1, w \in \mathcal{S}_2$ s.t. $\phi(v) = w$, $\mathcal{T} \models \lambda^*_{[\mathcal{S}_2]}(w) \equiv \lambda^*_{[\mathcal{S}_1]}(v)$,*
2. ***Plugin*** *($\mathcal{T}_{\sqsubseteq}$)-isomorphism, when $\forall v \in \mathcal{S}_1, w \in \mathcal{S}_2$ s.t. $\phi(v) = w$, $\mathcal{T} \models \lambda^*_{[\mathcal{S}_2]}(w) \sqsubseteq \lambda^*_{[\mathcal{S}_1]}(v)$,*
3. ***Subsume*** *($\mathcal{T}_{\sqsupseteq}$)-isomorphism, when $\forall v \in \mathcal{S}_1, w \in \mathcal{S}_2$ s.t. $\phi(v) = w$, $\mathcal{T} \models \lambda^*_{[\mathcal{S}_2]}(w) \sqsupseteq \lambda^*_{[\mathcal{S}_1]}(v)$,*

where:

$$\lambda^*_{[\mathcal{S}_1]}(v) = \bigsqcap \lambda_{[\mathcal{S}_1]}(v) \sqcap \bigsqcap_{\substack{\langle v,x \rangle \in \mathcal{E}_1, \\ x \notin \mathcal{S}_1}} \exists \lambda_{\mathcal{E}_1}(\langle v, x \rangle).C_{T_1(x)}, \tag{12}$$

and

$$\lambda^*_{[\mathcal{S}_2]}(w) = \bigsqcap \lambda_{[\mathcal{S}_2]}(w) \sqcap \bigsqcap_{\substack{\langle w,y \rangle \in \mathcal{E}_2, \\ y \notin \mathcal{S}_2}} \exists \lambda_{\mathcal{E}_2}(\langle w, y \rangle).C_{T_2(y)}, \tag{13}$$

Definition 12 states that for a given weak isomorphism between two $\rho_{\subseteq}$-subtrees, the vertices and edges that have induced a subtree will adjust the labels that provide the semantic layer. This adjustment allows for abduction of complex concept expressions which include concepts, role restrictions, or a combination of both.

Same as before, we distinguish three types of $\mathcal{T}$-isomorphisms between $\rho_{\subseteq}$-subtrees, that correspond to the exact, plugin, and subsume matching relations. To exemplify an isomorphism between $\rho_{\subseteq}$-subtrees of description trees and abduction of both concepts and role restrictions, consider the following example:

Example 2. *Let $\mathcal{T}$ be a TBox, such that $\mathcal{T} = \{B_2 \sqsubseteq A_2\}$ and let*

$$\begin{aligned} C &= A_0 \sqcap \exists r.A_1 \sqcap \exists s.A_2 \sqcap \exists p.A_3 \\ D &= B_0 \sqcap \exists r.B_1 \sqcap \exists s.B_2 \end{aligned} \tag{14}$$

be two concepts defined w.r.t. $\mathcal{T}$, such that $\mathcal{T} \not\models D \sqsubseteq C$. The description trees of C and D are shown in Figure 2.

It is apparent that the description trees T_C and T_D are not isomorphic, because there does not exist a mapping that preserves the structure between the trees. However, we can find subparts of T_C that are isomorphic to T_D. If we consider the following induced $\rho_{\subseteq}$-subtree of T_C:

$$\begin{aligned} T_C[\mathcal{S}_C] =& \langle \mathcal{S}_C, \mathcal{E}_{[\mathcal{S}_C]}, v_0, \lambda_{[\mathcal{S}_C]}, \lambda_{\mathcal{E}_{[\mathcal{S}_C]}} \rangle = \\ & \langle \mathcal{S}_C = \{v_0, v_1, v_2\}, \mathcal{E}_{[\mathcal{S}_C]} = \{\langle v_0, v_1 \rangle, \langle v_0, v_2 \rangle\}, v_0, \\ & \lambda_{[\mathcal{S}_C]} = \{v_0 \mapsto \{A_0\}, v_1 \mapsto \{A_1\}, v_2 \mapsto \{A_2\}, \\ & \lambda_{\mathcal{E}_{[\mathcal{S}_C]}} = \{\langle v_0, v_1 \rangle \mapsto r, \langle v_0, v_2 \rangle \mapsto s\}\rangle, \end{aligned} \tag{15}$$

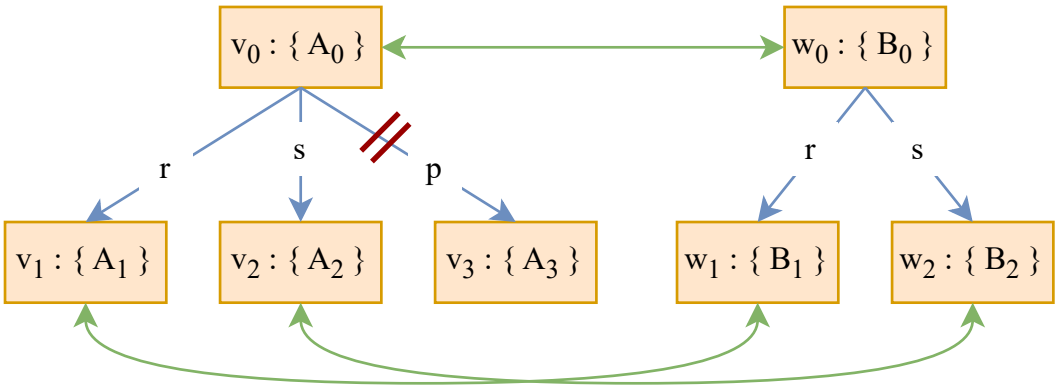

Fig. 2. Description trees T_C (left) and T_D (right) and a subtree isomorphism.

obtained by pruning the edge $\langle v_0, v_3 \rangle$, we can find a weak isomorphism $\phi : T_C[\mathcal{S}_C] \mapsto T_D$, such that $\phi = (v_0w_0)(v_1w_1)(v_2w_2)$. However, ϕ is missing a semantic layer in order to become a $\mathcal{T}_{\sqsubseteq}$-isomorphism and capture the concept inclusion between C and D. Currently, it holds that:

- $\mathcal{T} \models \lambda_{\mathcal{V}_D}(w_2) \sqsubseteq \lambda^*_{[\mathcal{S}_C]}(v_2)$, *i.e.* $\mathcal{T} \models B_2 \sqsubseteq A_2$.

On the other hand, we have that:

- $\mathcal{T} \not\models \lambda_{\mathcal{V}_D}(w_0) \sqsubseteq \lambda^*_{[\mathcal{S}_C]}(v_0)$, *i.e.* $\mathcal{T} \not\models B_0 \sqsubseteq A_0 \sqcap \exists p.A_3$, *and*
- $\mathcal{T} \not\models \lambda_{\mathcal{V}_D}(w_1) \sqsubseteq \lambda^*_{[\mathcal{S}_C]}(v_1)$, *i.e.* $\mathcal{T} \not\models B_1 \sqsubseteq A_1$.

Same as before, we can use this weak isomorphism and the "missing" relations to construct a hypothesis:

$$\mathcal{H} = \{B_1 \sqsubseteq A_1, B_0 \sqsubseteq A_0 \sqcap \exists p.A_3\},$$

which extends ϕ to a $(\mathcal{T} \cup \mathcal{H})_{\sqsubseteq}$-isomorphism. We now have $\mathcal{T} \cup \mathcal{H} \not\models C \sqcap D \equiv \bot$ and $\mathcal{T} \cup \mathcal{H} \models D \sqsubseteq C$.

The final **Point 3**, when performing abduction using subtree isomorphisms, puts significance on obtaining the maximal amount of knowledge that can be related between matching concept definitions. In other words, when performing abduction, we should not unnecessarily omit parts of definitions that already preserve the structure. This point addresses the parsimony of the abduction process.

Naturally, hypotheses that contain more complex expressions are harder to interpret in comparison to those containing simpler abductions. Moreover, for an abduction problem defined as in Definition 6, any arbitrary abduction up to the entire descriptions of the matching concepts can be used to explain the non-entailment. To this end, we introduce a minimality criteria which filters out hypotheses that contain unnecessary abductions of concepts and/or role restrictions.

To define minimal abductive solutions, we first impose a partial order on the isomorphisms.

Definition 13. *Let $T_1 = \langle \mathcal{V}_1, \mathcal{E}_1, v_0, \lambda_{\mathcal{V}_1}, \lambda_{\mathcal{E}_1} \rangle$ and $T_2 = \langle \mathcal{V}_2, \mathcal{E}_2, w_0, \lambda_{\mathcal{V}_2}, \lambda_{\mathcal{E}_2} \rangle$ be two description trees with $T_1[\mathcal{S}_1] \subseteq_\rho T_1$, $T_1[\mathcal{S}'_1] \subseteq_\rho T_1$, $T_2[\mathcal{S}_2] \subseteq_\rho T_2$, and $T_2[\mathcal{S}'_2] \subseteq_\rho T_2$. If $\phi : T_1[S_1] \mapsto T_1[S_2]$ and $\phi' : T_1[S'_1] \mapsto T_2[S'_2]$, then $\phi \preceq \phi'$ if:*

$$|\mathcal{S}_1| + |\mathcal{S}_2| \geqslant |\mathcal{S}'_1| + |\mathcal{S}'_2| \tag{16}$$

Definition 13 partially orders weak isomorphisms between $\rho_{\subseteq}$-subtrees, such that, the minimal element of the partially ordered set is the weak isomorphism obtained between largest $\rho^{\subseteq}$-subtrees. From this, we gain information

on how much the structure has changed in order to find isomorphic $\rho^{\sqsubseteq}$-subtrees of description trees. The more changes to the structure, the more complex restrictions are included in the hypotheses, making them more difficult to interpret. In cases where there are more than one isomorphic mappings, the minimal one is preferred. As a result of the hypotheses being constructed from weak isomorphisms, the partial order directly appoints minimal abductive solutions, i.e. hypotheses.

Definition 14. *Given an abduction problem $(\mathcal{T}, match_{\underline{\sqsubseteq}}(C,D))$, with $T_C = \langle \mathcal{V}_C, \mathcal{E}_C, v_0, \lambda_{\mathcal{V}_C}, \lambda_{\mathcal{E}_C} \rangle$ and $T_D = \langle \mathcal{V}_D, \mathcal{E}_D, w_0, \lambda_{\mathcal{V}_D}, \lambda_{\mathcal{E}_D} \rangle$ which represent the description trees of C and D, respectively, a solution $\mathcal{H}$ obtained from a weak isomorphism ϕ is considered minimal if there does not exist any $\mathcal{H}'$, obtained from ϕ', such that $\phi' \prec \phi$. We say that $\mathcal{H} \prec \mathcal{H}'$ if $\phi \prec \phi'$.*

We prioritize preserving the concept definitions, and, where necessary, find solutions obtained from minimal number of contractions to induce subtrees. To illustrate minimal solutions consider the following example.

Example 3. *Let $\mathcal{T}$ be a TBox and let C and D be two concept descriptions defined w.r.t. the signature of $\mathcal{T}$, such that $\mathcal{T} \not\models C \sqcap D \equiv \bot$ and $\mathcal{T} \not\models match_{\sqsubseteq}(C,D)$, where:*

$$C = A_0 \sqcap \exists r.A_1 \sqcap \exists s.A_2,$$

$$D = B_0 \sqcap \exists r.B_1 \sqcap \exists s.B_2.$$

The descriptions trees of the concepts C and D are shown in Figure 1. We can construct more than one hypotheses to explain this non-entailment. Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two hypotheses, such that:

$$\mathcal{H}_1 = \{A_0 \sqsubseteq B_0, A_1 \sqsubseteq B_1, A_2 \sqsubseteq B_2\},$$

$$\mathcal{H}_2 = \{A_0 \sqcap \exists r.A_1 \sqsubseteq B_0 \sqcap \exists r.B_1, A_2 \sqsubseteq B_2\}$$

Both $\mathcal{H}_1$ and $\mathcal{H}_2$ explain the non-entailment and we have $\mathcal{T} \cup \mathcal{H}_1 \models C \sqsubseteq D$ and $\mathcal{T} \cup \mathcal{H}_2 \models C \sqsubseteq D$. However, $\mathcal{H}_2$ is constructed by performing unnecessary contractions in the description trees, and, consequently the concept definitions, that only introduces more complex expressions to interpret in the hypothesis. In fact, we can go up to any arbitrary number of contractions that induce subtrees and even construct a hypothesis $\mathcal{H}_n = \{A_0 \sqcap \exists r.A_1 \sqcap \exists s.A_2 \sqsubseteq B_0 \sqcap \exists r.B_1 \sqcap \exists s.B_2\}$, which includes the complete definitions of the concepts C and D. However, this explanation does not provide any meaningful relations that could be used to explain the non-entailment and update the terminological knowledge. To this end, we filter the hypotheses w.r.t. the minimality criteria in Definition 14.

Isomorphisms between $\rho_{\sqsubseteq}$-subtrees allow for relating relevant concepts and role restrictions whilst preserving the structure of concept definitions. Furthermore, they do not restrict the process of abduction to a predetermined set of abducibles, but they allow for any form of a complex expression to be abducted. The only limitation is w.r.t. the signature of a background knowledge for which we want to explain a certain non-entailment. Moreover, the introduced minimality criteria filters hypotheses with arbitrary abductions and produces meaningful explanations.

5.1. Computing Subtree Isomorphisms

One of the most fundamental computational problems on trees is the subtree isomorphism problem, which asks whether a given tree is contained in another given tree. The subtree isomorphism problem has a few variants, which are mainly dependent on whether the trees are rooted or unrooted, whether the degrees of nodes are bounded, and whether an order on their nodes must be preserved. We accustom the subtree isomorphism problem for the special case of description trees to solve abduction problems in $\mathcal{EL}_\bot$.

Definition 15. *Let $T_1 = \langle \mathcal{V}_1, \mathcal{E}_1, v_0, \lambda_{\mathcal{V}_1}, \lambda_{\mathcal{E}_1} \rangle$ and $T_2 = \langle \mathcal{V}_2, \mathcal{E}_2, w_0, \lambda_{\mathcal{V}_2}, \lambda_{\mathcal{E}_2} \rangle$ be two description trees. A $\rho_{\sqsubseteq}$-subtree isomorphism is an isomorphism between a $\rho_{\sqsubseteq}$-subtree $T_1[\mathcal{S}_1] = \langle \mathcal{S}_1, \mathcal{E}_{\mathcal{S}_1}, v_0, \lambda_{[\mathcal{S}_1]}, \lambda_{\mathcal{E}_{\mathcal{S}_1}} \rangle$ of T_1 and a $\rho_{\sqsubseteq}$-subtree $T_2[\mathcal{S}_2] = \langle \mathcal{S}_2, \mathcal{E}_{\mathcal{S}_2}, w_0, \lambda_{\mathcal{S}_2}, \lambda_{\mathcal{E}_{\mathcal{S}_2}} \rangle$ of T_2, such that $T_1[\mathcal{S}_1] \cong T_2[\mathcal{S}_2]$.*

To compute subtree isomorphisms between description trees and derive hypotheses to explain non-entailments of semantic matching we partition the children of vertices in their respective trees w.r.t. the labels of edges connecting parents to their children. We define an *equivalence relation* on the set of children for a vertex v in a description tree.

Theorem 2. *Let $\sim$ be a relation on a non-empty set of children, $N(v)$, for a node $v \in \mathcal{V}$ of a tree $T = \langle \mathcal{V}, \mathcal{E}, v_0, \lambda_\mathcal{V}, \lambda_\mathcal{E} \rangle$, such that for $x, y \in N(v)$, $x \sim y$ iff $\langle v, x \rangle \in \mathcal{E}$, $\langle v, y \rangle \in \mathcal{E}$ and $\lambda_\mathcal{E}(\langle v, x \rangle) = \lambda_\mathcal{E}(\langle v, y \rangle)$. Then, $\sim$ is an equivalence relation.*

Proof. For $\sim$ to be an equivalence relation we need to show that it is reflexive, symmetric, and transitive. Let $T = \langle \mathcal{V}, \mathcal{E}, v_0, \lambda_\mathcal{V}, \lambda_\mathcal{E} \rangle$ and $v \in \mathcal{V}$ an arbitrary node.

Reflexive property: For $x \in N(v)$, we have that $\lambda_\mathcal{E}(\langle v, x \rangle) = \lambda_\mathcal{E}(\langle v, x \rangle)$.

Symmetric property: For $x, y \in N(v)$, if $\lambda_\mathcal{E}(\langle v, x \rangle) = \lambda_\mathcal{E}(\langle v, y \rangle)$, then $\lambda_\mathcal{E}(\langle v, y \rangle) = \lambda_\mathcal{E}(\langle v, x \rangle)$.

Transitive property: For $x, y, z \in N(v)$, if $\lambda_\mathcal{E}(\langle v, x \rangle) = \lambda_\mathcal{E}(\langle v, y \rangle)$ and $\lambda_\mathcal{E}(\langle v, y \rangle) = \lambda_\mathcal{E}(\langle v, z \rangle)$, then $\lambda_\mathcal{E}(\langle v, x \rangle) = \lambda_\mathcal{E}(\langle v, z \rangle)$.

Thus, $\sim$ is an equivalence relation. □

An equivalence relation defines exclusive classes whose members bear the relation to each other and not to those in other classes. The set of all equvalence classes is defined as:

$$N(v)_{/\sim} = \{[x] \mid x \in N(v)\}, \text{ where } [x] := \{y \in N(v) \mid x \sim y\}, \tag{17}$$

is the equivalence class of x.

Theorem 3. *Let $\sim$ be an equivalence relation on a non-empty set of neighbors $N(v)$, for a node $v \in \mathcal{V}$ of a tree $T = \langle \mathcal{V}, \mathcal{E}, v_0, \lambda_\mathcal{V}, \lambda_\mathcal{E} \rangle$. The collection of equivalence classes under $\sim$ forms a partition of $N(v)$.*

Proof. Let $\sim$ be an equivalence relation on $N(v)$ and let $x \in N(v)$.

1) We need to show that $N(v) = \bigcup_x [x]$, where $[x]$ is termed a *cell* (a subset of $N(v)$). If $x \in N(v)$, then x belongs to $[x]$ (by the reflexive property we have that $x \sim x$, therefore $x \in [x]$). Since this is true for each element of $N(v)$, we have that $N(v) = \bigcup_x [x]$.

2) We need to show that if $[x]$ and $[y]$ are any two cells, then either $[x] = [y]$ or $[x] \cap [y] = \emptyset$ holds.

2.1) Let $\sim$ be an equivalence relation defined on $N(v)$ with $x, y \in N(v)$ and let $x \in [y]$. Then $[x] = [y]$. If $x \in [y]$ then by definition of equivalence classes we have $x \sim y$. First we show that $[x] \subseteq [y]$. Let $a \in [x]$. By definition of equivalence classes, $a \sim x$, and by the transitive property we have that $a \sim y$. Therefore, $a \in [y]$ and $[x] \subseteq [y]$. Next, we show that $[y] \subseteq [x]$. If $b \in [y]$, then $b \sim y$. By symmetry we have that $y \sim x$ and by transitivity that $b \sim x$. Thus, $b \in [x]$ and $[y] \subseteq [x]$. Thus we have that $[x] = [y]$ if $x \in [y]$.

2.2) Suppose $[x] \cap [y] \neq \emptyset$. Then, there must be some element a, such that $a \in [x]$ and $a \in [y]$. Then, $[a] = [x]$ and $[a] = [y]$. Thus, $[x] = [y]$. □

Once the children are partitioned, vertices that belong in an equivalence class in one tree can be mapped to vertices that belong in an equivalence class in the other tree, such that the equivalence classes in the respective trees are obtained from same edge labels. Let $T_1 = \langle \mathcal{V}_1, \mathcal{E}_1, v_0, \lambda_{\mathcal{V}_1}, \lambda_{\mathcal{E}_1} \rangle$ and $T_2 = \langle \mathcal{V}_2, \mathcal{E}_2, w_0, \lambda_{\mathcal{V}_2}, \lambda_{\mathcal{E}_2} \rangle$ be two description trees. We define the relation:

$$\mu(v, w) \subseteq N(v)_{/\sim} \times N(w)_{/\sim}, \tag{18}$$

for nodes $v \in \mathcal{V}_1$ and $w \in \mathcal{V}_2$, such that:

$$\mu(v, w) := \{\langle [x], [y] \rangle \mid x \in N(v), y \in N(w), \langle v, x \rangle \in \mathcal{E}_1, \langle w, y \rangle \in \mathcal{E}_2 \text{ and } \lambda_{\mathcal{E}_1}(\langle v, x \rangle) = \lambda_{\mathcal{E}_2}(\langle w, y \rangle)\} \tag{19}$$

The relation $\mu(v, w)$ pairs the equivalence classes that contain children of v and w which come from partitions with equal edge labels. To illustrate this, consider the small description trees in Figure 3. The children of v_0 and w_0 are partitioned in three equivalence classes in their respective trees, i.e. $[v_1] = \{v_1, v_2\}$, $[v_3] = \{v_3\}$, and $[v_4] = \{v_4\}$

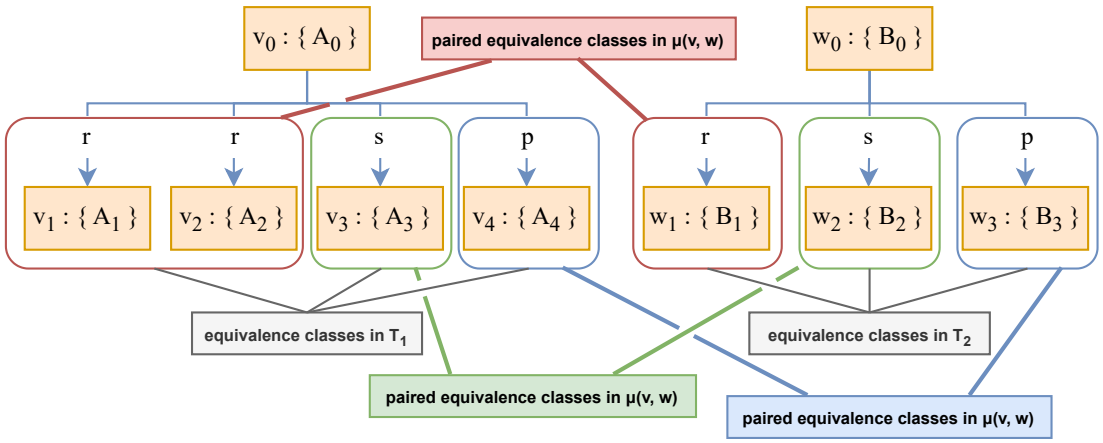

Fig. 3. Example description trees T_1 (left) and T_2 (right), partitions of the children of the roots in equivalence classes, and paired equivalence classes between the trees.

in T_1, and $[w_1] = \{w_1\}$, $[w_2] = \{w_2\}$, and $[w_3] = \{w_3\}$ in T_2. The relation $\mu(v_0, w_0)$ will contain all pairs, such that $\mu(v_0, w_0) = \{\langle [v_1], [w_1]\rangle, \langle [v_3], [w_2]\rangle, \langle [v_4], [w_3]\rangle\}$ and will conveniently pair sets of nodes that when mapped will satisfy condition 2 of Definition 8, i.e. will preserve the edges and their labels in a mapping.

Finally, we map the partitioned nodes from the respective trees using the relation $\mu(v, w)$. This is done for all vertices that preserve the structure between the trees, whilst the vertices from partitions that cannot be mapped are contracted, thus inducing a subtree.

To find all pairs of mapped nodes that preserve the structure, we compute *injective maps between distinct subsets of the equivalence classes in* $\mu(v, w)$.

Let A and B be arbitrary sets, such that $|A| \leqslant |B|$. We define the injective function between A and B as $i : A \overset{1-1}{\mapsto} B$, such that for $a, b \in A$, $i(a) \neq i(b) \implies a \neq b$. We use the notation $A^{\underline{B}}$ for the set of all injections from A to B.

Proposition 4. *Given two non-empty sets A and B, if there is an injective mapping* $i : A \overset{1-1}{\mapsto} B$ *then* $|A| \leqslant |B|$.

Proof. We prove this by using Dirichlet's principle (a.k.a. the pigeonhole principle).

Consider a mapping $i : A \overset{1-1}{\mapsto} B$ and let $|A| = n$ and $|B| = m$. Then, n elements of A are paired with m elements of B. If $n > m$, then at least one element of A must be paired with more than one element of B. Since an injective mapping is a 1-1, i.e. one element from A must be paired with only one element from B, then it follows that the cardinality of A must be strictly smaller or equal to the cardinality of B, i.e. $|A| \leqslant |B|$. □

Corollary 5. *Given two non-empty sets A and B, if* $|A| \leqslant |B|$*, then we can construct an injective mapping* $i : A \overset{1-1}{\mapsto} B$.

Proof. The proof is ones again done by using Dirichlet's principle.

Consider $|A| = n$ and $|B| = m$, such that $n \leqslant m$. By Dirichlet's principle when we pair n elements of A with m elements of B, such that $n \leqslant m$, we can pair each unique element of A with a unique element of B, i.e. in a one-to-one correspondence. Thus, we have constructed a map $i : A \overset{1-1}{\mapsto} B$, i.e. injective. □

Going back to our equivalence classes in $\mu(v, w)$, dependent on the cardinality of $[x]$ and $[y]$, we can either find the set of all injections:

a) $[x]^{\underline{[y]}}$ for $|[x]| \leqslant |[y]|$, or
b) $[y]^{\underline{[x]}}$ for $|[x]| > |[y]|$.

In any case, the injective mappings will be between subsets of $[x]$ to $[y]$ or vice versa and the mapped sets will have equal cardinalities. To illustrate this, consider the following equivalence classes:

- $[x] = \{v_1, v_2, v_3\}$, and
- $[y] = \{w_1, w_2\}$.

We can find injective mappings from $[y]$ to $[x]$, since $|[y]| < |[x]|$. Then, the set of all injective mappings is:

$$[y]^{\underline{[x]}} = \{\{(w_1, v_1), (w_2, v_2)\}, \{(w_1, v_1), (w_2, v_3)\}, \{(w_1, v_2), (w_2, v_1)\}, \{(w_1, v_2), (w_2, v_3)\}, \{(w_1, v_3), (w_2, v_1)\}, \{(w_1, v_3), (w_2, v_2)\}\}.$$

Since $|[x]| > |[y]|$, to construct injective mappings we map $|[y]|$ nodes in $[y]$ to $|[y]|$ nodes in $[x]$ in a 1-1 correspondence. Thus, some elements in $[x]$ will not be mapped to. These nodes will be contracted and will induce a subtree in their respective tree. Since we omit some nodes in $[x]$, because there are no unique elements from $[y]$ to map them to in a 1-1 correspondence, essentially the search for injective mappings is brought down between $[y]$ and a subset of $[x]$ that is of size $|[y]|$. Because isomorphisms preserve the trees' structures, the number of nodes is also preserved up to isomorphism, and if we have equivalence classes with different sizes, we need to induce a subtree in the tree that has a surplus of nodes. This will identify and filter nodes that cannot preserve the structure of the trees because the trees are of different sizes. Each injective mapping represents one abductive solution and consequently to obtain all abductive solutions all injective mappings are obtained.

Proposition 6. *Let A and B be arbitrary sets, such that $|A| = |B|$, and let $f : A \mapsto B$ be an injection. Then, f is also a bijection.*

Proof. For f to be a bijection, f needs to be injective and surjective.

Let A and B be arbitrary sets, such that $|A| = n$, $|B| = m$, and $n = m$, $f : A \mapsto B$ is already injective. By definition, a function is injective such that for $a, a' \in A$, if $a \neq a'$, then $f(a) \neq f(a')$. Thus, we have that $n = m$ and n elements in A that are paired with m elements in B, such that for each element in A there is a unique element in B.

By definition a function $s : X \mapsto Y$ is said to be surjective if $\forall y \in Y, \exists x \in X$, s.t. $s(x) = y$. Since f is injective, it holds that $\forall a \in A, \exists b \in B$, $f(a) = b$. Thus, f is also surjective.

Since f is injective and surjective, then f is also bijective. □

Since the subsets of $[x]$ have cardinality $|[y]|$, as a consequence of Proposition 6 the obtained injective mappings are also bijective in nature. Since we are obtaining bijective mappings, the order between the pairs of mapped nodes does not matter. Hence, we denote the set of all injective (also bijective) mappings between equivalence classes by $[x]^{\underline{[y]}}$, with which we obtain all combinations of mappings between nodes that preserve the structure of the trees.

For each pair of equivalence classes obtained from the equivalence relation, we find the set of all injective maps, such that:

$$I(v, w) = \{[x]^{\underline{[y]}} \mid \langle [x], [y] \rangle \in \mu(v, w)\} \tag{20}$$

Each set of injections is unique to the respective pair of equivalence classes. To obtain all combinations of mapped vertices that produce isomorphisms, we search for the Cartesian product between the distinct sets of injections:

$$\mathcal{M}(v, w) = \{\bigcup S_i \mid S_i \in \prod_{i \in I(v,w)} i\} \tag{21}$$

To illustrate how we find equivalence classes and how we generate the sets $I(v, w)$ and $\mathcal{M}(v, w)$ we construct a small example:

Example 4. *Consider the description trees in Figure 4.*

We start from the roots of T_1 and T_2 and find $N(v_0)$ and $N(w_0)$, which are:

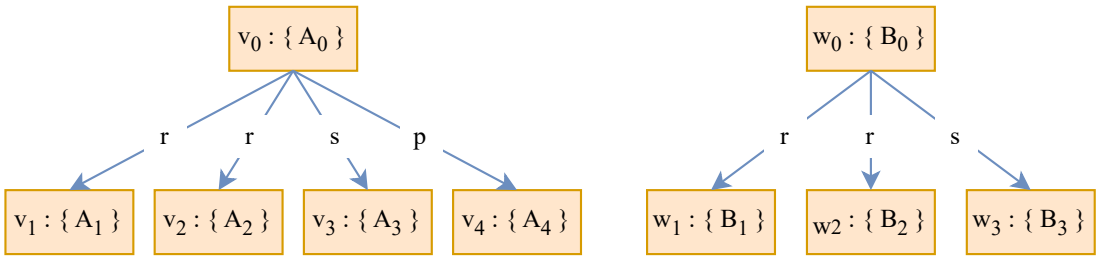

Fig. 4. Example description trees T_1 (left) and T_2 (right).

- $N(v_0) = \{v_1, v_2, v_3, v_4\}$,
- $N(w_0) = \{w_1, w_2, w_3\}$.

Next, we partition $N(v_0)$ and $N(w_0)$ w.r.t. Eq. 17 and get:

- $N(v_0)_{/\sim} = \{[v_1], [v_3], [v_4]\}$,
- $N(w_0)_{/\sim} = \{[w_1], [w_3]\}$,

where $[v_1] = \{v_1, v_2\}$, $[v_3] = \{v_3\}$, $[v_4] = \{v_4\}$, $[w_1] = \{w_1, w_2\}$, and $[w_3] = \{w_3\}$. We can already see from the trees in Figure 4 how the edge labels constrain the structure of the trees. For example, the node v_4 in T_1 cannot be mapped to any node in T_2, because it is connected by an edge that is labeled with p in T_1, and there does not exist an edge with the same label in T_2. Hence, this part of the structure cannot be preserved and we need to prune it and construct an induced subtree of T_1.

From here, we obtain $\mu(v_0, w_0)$ w.r.t. Eq. 19 and get:

- $\mu(v_0, w_0) = \{\langle [v_1], [w_1] \rangle, \langle [v_3], [w_3] \rangle\}$,

in which we can see that the partition w.r.t. the label p in T_1 is not included, since it cannot preserve the structure between the trees. Now we can find the set of all mappings. From $\mu(v_0, w_0)$ we know that we need to find $[v_1]^{\underline{[w_1]}}$ and $[v_3]^{\underline{[w_3]}}$:

- $[v_1]^{\underline{[w_1]}} = \{\{\langle v_1, w_1 \rangle, \langle v_2, w_2 \rangle\}, \{\langle v_1, w_2 \rangle, \langle v_2, w_1 \rangle\}\}$, *and*
- $[v_3]^{\underline{[w_3]}} = \{\{\langle v_3, w_3 \rangle\}\}$,

which are included in $I(v_0, w_0)$:

- $I(v_0, w_0) = \{\{\{\langle v_1, w_1 \rangle, \langle v_2, w_2 \rangle\}, \{\langle v_1, w_2 \rangle, \langle v_2, w_1 \rangle\}\}, \{\{\langle v_3, w_3 \rangle\}\}\}$.

We can see in $I(v_0, w_0)$ all combinations of injective (also bijective) mappings for the respective equivalence classes. To also find all combinations of isomorphic mappings between all equivalence classes, we formulate $\mathcal{M}(v_0, w_0)$ w.r.t. Eq. 21 and get:

- $\mathcal{M}(v_0, w_0) = \{\bigcup S_i \mid S_i \in [v_1]^{\underline{[w_1]}} \times [v_3]^{\underline{[w_3]}}\} = \{\{\langle v_1, w_1 \rangle, \langle v_2, w_2 \rangle, \langle v_3, w_3 \rangle\}, \{\langle v_1, w_2 \rangle, \langle v_2, w_1 \rangle, \langle v_3, w_3 \rangle\}\}$

$\mathcal{M}(v_0, w_0)$ contains the bijective mappings between the children of v_0 in T_1 and the children of w_0 in T_2. Each mapping is obtained by first partitioning the children of v_0 and w_0 w.r.t. the edge labels in their trees. The approach for generating $\mathcal{M}(v_0, w_0)$ makes sure that the sets of mapped children are obtained from sets with corresponding edge labels. Further, this is done iteratively for all nodes that are mapped thus far and the result will yield all subtree isomorphisms between trees that have larger depths.

Notice that $\mathcal{M}(v, w)$ could scale by a very large factor, which is dependent on the sizes of the equivalence classes. For some $[x]$ and $[y]$, for which we want to find $[x]^{\underline{[y]}}$, we have that $|[x]^{\underline{[y]}}| = \frac{|[x]|!}{(|[x]|-|[y]|)!}$. This is a direct consequence of how the matching concepts are defined for the abduction problem. Because the equivalence classes are obtained by partitioning the nodes w.r.t. the edge labels, large sizes of equivalence classes can occur when there is a repetition

of edge labels, i.e. we have a high number of repetitive roles in the matching concepts. Thus, a rich terminological knowledge that introduces a variety of roles that can be used in the definitions of concepts will improve the scalability of our method. Furthermore, our approach could be optimized to include not only concept, but also role inclusions, by taking the role hierarchy of the background knowledge into perspective. This could present the option for equivalence classes to be combined or separated and could potentially enhance the method's performance.

To obtain isomorphic mappings, we construct a *Solutions Tree*.

Definition 16. *A Solutions Tree is a node-labeled rooted tree* $T_\Phi = \langle \mathcal{V}_\Phi, \mathcal{E}_\Phi, \phi_0, \lambda_\Phi \rangle$ *that contains vertex mappings between two description trees* $T_1 = \langle \mathcal{V}_1, \mathcal{E}_1, v_0, \lambda_{\mathcal{V}_1}, \lambda_{\mathcal{E}_1} \rangle$ *and* $T_2 = \langle \mathcal{V}_2, \mathcal{E}_2, w_0, \lambda_{\mathcal{V}_2}, \lambda_{\mathcal{E}_2} \rangle$. *The Solutions Tree contains nodes* $\phi \in \mathcal{V}_\Phi$, *such that:*

$$\lambda_\Phi(\phi) \subseteq \{\langle v, w \rangle \mid v \in \mathcal{V}_1, w \in \mathcal{V}_2\} \tag{22}$$

Each unique path from the root to a leaf node in the solutions tree contains a unique subtree isomorphism between T_1 *and* T_2. *The isomorphisms are returned from the set of complete paths in the solutions tree. If the Solutions Tree contains only its root,* ϕ_0, *then we refer to it as the trivial solution.*

The solutions tree is a rooted tree, which contains nodes that carry mapped vertices between two $\rho_\subseteq$-subtrees. Each unique complete path in the solutions tree provides a unique combination of mapped nodes between isomorphic $\rho_\subseteq$-subtrees of some description trees. We inductively define the sets $\mathcal{V}_\Phi$ and $\mathcal{E}_\Phi$ and the node labels of the solutions tree.

$$\begin{aligned}
&\textbf{B1}\ \ \phi_0 \in T_\phi, \lambda_\Phi = \{\phi_0 \mapsto \{\langle v_0, w_0 \rangle\}\} \\
&\textbf{R1}\ \text{if}\ \phi \in \mathcal{V}_\Phi,\ \text{then for all}\ m \in \prod_{\langle v,w \rangle \in \lambda_\Phi(\phi)} \mathcal{M}(v,w), \phi^* \in \mathcal{V}_\Phi\ \text{and}\ \langle \phi, \phi^* \rangle \in \mathcal{E}_\Phi\ \text{where}\ \lambda_\Phi(\phi^*) = m. \\
&\textbf{R2}\ \ \text{Nothing else is in}\ \mathcal{V}_\Phi\ \text{and}\ \mathcal{E}_\Phi.
\end{aligned} \tag{23}$$

The set of all isomorphic mappings Φ is then obtained from the set of all complete paths Π^0 in the solutions tree T_Φ as follows:

$$\Phi = \{\bigcup_{\phi \in \pi} \lambda_\Phi(\phi) \mid \pi \in \Pi^0\} \tag{24}$$

Claim 1. *For all* $\phi \in \Phi$, *the following proposition* $P(\phi)$ *is true: there exist* $\rho_\subseteq$*-subtrees* $T_1[\mathcal{S}_1] \subseteq_\rho T_1$ *and* $T_2[\mathcal{S}_2] \subseteq_\rho T_2$, *where* $\mathcal{S}_1 \subseteq_\rho \mathcal{V}_1$ *and* $\mathcal{S}_2 \subseteq_\rho \mathcal{V}_2$ *such that* ϕ *is a weak isomorphism, between* $T_1[\mathcal{S}_1]$ *and* $T_2[\mathcal{S}_2]$

Proof. We prove this by structural induction on the induced subtrees $T_1[\mathcal{S}_1]$ and $T_2[\mathcal{S}_2]$.

Let $T_1 = \langle \mathcal{V}_1, \mathcal{E}_1, v_0, \lambda_{\mathcal{V}_1}, \lambda_{\mathcal{E}_1} \rangle$ and $T_2 = \langle \mathcal{V}_2, \mathcal{E}_2, w_0, \lambda_{\mathcal{V}_2}, \lambda_{\mathcal{E}_2} \rangle$ be two description trees.

Basis: The solutions tree $T_\Phi = \langle \mathcal{V}_\Phi, \mathcal{E}_\Phi, \phi_0, \lambda_\Phi \rangle$, where $\mathcal{V}_\Phi = \{\phi_0\}, \mathcal{E}_\Phi = \emptyset, \lambda_\Phi = \{\phi_0 \mapsto \{\langle v_0, w_0 \rangle\}\}$, contains only its root node ϕ_0. Then, the set of all complete paths in the solutions tree is $\Pi^0 = \{\langle \phi_0 \rangle\}$, from which we get a mapping $\phi_0 = \{\langle v_0, w_0 \rangle\} \in \Phi$, such that $\phi_0(v_0) = w_0$. Thus, there exist $\rho_\subseteq$-subtrees:

$$\begin{aligned}
&T_1[\mathcal{S}_1] = \langle \mathcal{S}_1 = \{v_0\}, \mathcal{E}_{[\mathcal{S}_1]} = \emptyset, v_0, \lambda_{[\mathcal{S}_1]}, \lambda_{\mathcal{E}_{[\mathcal{S}_1]}} \rangle,\ \text{and} \\
&T_2[\mathcal{S}_2] = \langle \mathcal{S}_2 = \{w_0\}, \mathcal{E}_{[\mathcal{S}_2]} = \emptyset, w_0, \lambda_{[\mathcal{S}_2]}, \lambda_{\mathcal{E}_{[\mathcal{S}_2]}} \rangle
\end{aligned} \tag{25}$$

which are induced by subsets $\mathcal{S}_1 = \{v_0\}$ for T_1 and $\mathcal{S}_2 = \{w_0\}$ for T_2. Since the induced $\rho_\subseteq$-subtrees contain only the roots, which are mapped, by definition ϕ_0 is a weak isomorphism. Thus, $P(\phi_0)$ is true.

Induction: We need to show that $P(\phi_i)$ is true, such that $\phi_i = \phi_0 \cup m_i$, for all $m_i \in \mathcal{M}(v_0, w_0)$. To do this, we need to follow the formulation of $m_i \in \mathcal{M}(v_0, w_0)$. We have two cases:

1. If we do not extend the induced subtrees $T_1[\mathcal{S}_1]$ and $T_2[\mathcal{S}_2]$, then the induced subtrees once again contain only the roots and their sets $N(v_0)$ and $N(w_0)$ are empty. Hence, we do not have any children of v_0 and w_0 to partition, for all $m_i \in \mathcal{M}(v_0, w_0), m_i = \emptyset$. Thus, $\phi_i = \phi_0$, which from the base case, $P(\phi_i) = P(\phi_0)$ is true.

2. If we extend the T_1 and T_2, such that:

$$T_C = \langle \{v_0\} \cup \{v_i \mid 1 \leqslant i \leqslant n\}, \{\langle v_0, v_i\rangle \mid 1 \leqslant i \leqslant n\}, v_0, \lambda_{\mathcal{V}_C}, \lambda_{\mathcal{E}_C}\rangle,$$
$$T_D = \langle \{w_0\} \cup \{w_i \mid 1 \leqslant i \leqslant m\}, \{\langle w_0, w_i\rangle \mid 1 \leqslant i \leqslant m\}, w_0, \lambda_{\mathcal{V}_D}, \lambda_{\mathcal{E}_D}\rangle,$$

then we have nodes in $N(v_0)$ and $N(w_0)$ that we can partition w.r.t. the edge labels. We can obtain $\mathcal{M}(v_0, w_0)$ from $\mu(v_0, w_0)$ and $I(v_0, w_0)$. For each $m_i \in \mathcal{M}(v_0, w_0), m_i \neq \emptyset$, and for each $\langle x, y\rangle \in m_i$, we have $\lambda_{\mathcal{E}_1}(\langle v_0, x\rangle) = \lambda_{\mathcal{E}_2}(\langle w_0, y\rangle)$.

Hence, all vertices $x \in \mathcal{V}_1$ and $y \in \mathcal{V}_2$, for which $\lambda_{\mathcal{E}_1}(\langle v_0, x\rangle) \neq \lambda_{\mathcal{E}_2}(\langle w_0, y\rangle)$, are omitted from each $m_i \in \mathcal{M}(v_0, w_0)$, we have new $\rho_{\subseteq}$-subtrees that are induced from those vertices, i.e. $T_1[\mathcal{S}_1]$ and $T_2[\mathcal{S}_2]$ for $\mathcal{S}_1 \subseteq \mathcal{V}_1$ and $\mathcal{S}_2 \subseteq \mathcal{V}_2$:

$$T_1[\mathcal{S}_1] = \langle \mathcal{S}_1 = \{v_0\} \cup \{v_i \mid 1 \leqslant i \leqslant k\}, \mathcal{E}_{[\mathcal{S}_1]} = \{\langle v_0, v_i\rangle \mid 1 \leqslant i \leqslant k\}, v_0, \lambda_{[\mathcal{S}_1]}, \lambda_{\mathcal{E}_{[\mathcal{S}_1]}}\rangle, \text{ and}$$
$$T_2[\mathcal{S}_2] = \langle \mathcal{S}_2 = \{w_0\} \cup \{w_i \mid 1 \leqslant i \leqslant k\}, \mathcal{E}_{[\mathcal{S}_2]} = \{\langle w_0, w_i\rangle \mid 1 \leqslant i \leqslant k\}, w_0, \lambda_{[\mathcal{S}_2]}, \lambda_{\mathcal{E}_{[\mathcal{S}_2]}}\rangle, \tag{26}$$

From Proposition 6 it follows that each $m_i \in \mathcal{M}(v_0, w_0)$ is a bijective mapping $m_i : \mathcal{S}_1 \setminus \{v_0\} \mapsto \mathcal{S}_2 \setminus \{w_0\}$, where for all $x \in \mathcal{S}_1$ and $y \in \mathcal{S}_2$ such that $m_i(x) = y$, we have $\lambda_{\mathcal{E}_{[\mathcal{S}_1]}}(\langle v_0, x\rangle) = \lambda_{\mathcal{E}_{[\mathcal{S}_2]}}(\langle w_0, y\rangle)$ in the $\rho_{\subseteq}$-subtrees, and $\lambda_{\mathcal{E}_1}(\langle v_0, x\rangle) = \lambda_{\mathcal{E}_2}(\langle w_0, y\rangle)$ in the original trees (condition 2 of Definition 8 for weak isomorphisms is satisfied).

Since each $m_i \in \mathcal{M}(v_0, w_0)$ contains mapped children of v_0 and w_0 and is a bijective mapping between the subsets $\mathcal{S}_1 \setminus \{v_0\}$ and $\mathcal{S}_2 \setminus \{w_0\}$, m_i is an extension of ϕ_0, and we have $\phi_i = \phi_0 \cup m_i : \mathcal{S}_1 \mapsto \mathcal{S}_2$ for each $m_i \in \mathcal{M}(v_0, w_0)$, and for all $v \in \mathcal{S}_1, w \in \mathcal{S}_2$, $\phi_i(v) = w$ and it includes the mapped roots, i.e. $\phi_i(v_0) = (w_0)$ (condition 1 of Definition 8 for weak isomorphisms is satisfied). Thus, we have that there exist some $\rho_{\subseteq}$-subtrees, for which each ϕ_i is a weak isomorphisms between them, i.e. $P(\phi_i)$ is true.

Assuming for some $\phi \in \Phi$ that $P(\phi)$ is true, i.e. there exist some $\rho_{\subseteq}$-subtrees $T_1[\mathcal{S}_1]$ and $T_2[\mathcal{S}_2]$, such that ϕ is a weak isomorphism between them, for any $v \in \mathcal{S}_1$ and $w \in \mathcal{S}_2$, such that $\phi(v) = w$, where $T_1(v)$ is the subtree in T_1 rooted in v and $T_2(w)$ is the subtree in T_2 rooted in w. In other words, we extend the $\rho_{\subseteq}$-subtrees $T_1[\mathcal{S}_1]$ and $T_2[\mathcal{S}_2]$ with the children of $v \in T_1$ and $w \in T_2$. Thus, we get:

$$T_1[\mathcal{S}'_1] = \langle \mathcal{S}'_1 = \mathcal{S}_1 \cup \{v_i \mid 1 \leqslant i \leqslant n\}, \mathcal{E}_{[\mathcal{S}'_1]} = \{\langle v, v_i\rangle \mid 1 \leqslant i \leqslant n\}, v, \lambda_{[\mathcal{S}'_1]}, \lambda_{\mathcal{E}_{[\mathcal{S}'_1]}}\rangle, \text{ and}$$
$$T_2[\mathcal{S}'_2] = \langle \mathcal{S}'_2 = \mathcal{S}_2 \cup \{w_i \mid 1 \leqslant i \leqslant m\}, \mathcal{E}_{[\mathcal{S}'_2]} = \{\langle w, w_i\rangle \mid 1 \leqslant i \leqslant m\}, w, \lambda_{[\mathcal{S}'_2]}, \lambda_{\mathcal{E}_{[\mathcal{S}'_2]}}\rangle, \tag{27}$$

Similar, as previously when the extension was started from the mapped roots, we obtain $\mathcal{M}(v, w)$, which contain bijective mappings between the sets $\mathcal{S}'_1 \setminus \{v\}$ and $\mathcal{S}'_2 \setminus \{w\}$. Thus, each obtained $m \in \mathcal{M}(v, w)$ is an extension of ϕ. As shown previously, for each $m \in \mathcal{M}(v, w)$, $\phi \cup m$ is a weak isomorphism and since $P(\phi)$ is true, we have that $P(\phi \cup m)$ is true, i.e. there exist $\rho_{\subseteq}$-subtrees $T_1[\mathcal{S}'_1]$ and $T_2[\mathcal{S}'_2]$ for which its extension $\phi \cup m$ is a weak isomorphism. □

For two given $\rho_{\subseteq}$-subtrees $T_1[\mathcal{S}_1] = \langle \mathcal{S}_1, \mathcal{E}_{[\mathcal{S}_1]}, v_0, \lambda_{\mathcal{S}_1}, \lambda_{\mathcal{E}_{[\mathcal{S}_1]}}\rangle$ and $T_2[\mathcal{S}_2] = \langle \mathcal{S}_2, \mathcal{E}_{[\mathcal{S}_2]}, w_0, \lambda_{\mathcal{S}_2}, \lambda_{\mathcal{E}_{[\mathcal{S}_2]}}\rangle$, the hypotheses are then formulated from the set of all isomorphic mappings Φ in the following way: for both trees find the vertices that have been contracted and update the vertex labels w.r.t. Definition 12. For example, for a given label of an arbitrary node x in a $\rho_{\subseteq}$-subtree $T[\mathcal{S}] = \langle \mathcal{S}, \mathcal{E}_{[\mathcal{S}]}, v_0, \lambda_{[\mathcal{S}]}, \lambda_{\mathcal{E}_{[\mathcal{S}]}}\rangle$, the new label of x is:

$$\lambda^*_{[\mathcal{S}]}(x) = \bigsqcap_{x \in \mathcal{S}} \lambda_{[\mathcal{S}]}(x) \sqcap \bigsqcap_{\langle x, y\rangle \in \mathcal{E}, y \notin \mathcal{S}} \exists \lambda_{\mathcal{E}}(\langle x, y\rangle).C_{T(y)}, \tag{28}$$

Finally, for some $\rho_{\subseteq}$-subtree isomorphisms in Φ, the hypothesis $\mathcal{H}$ is constructed as follows:

$$\mathcal{H} = \{\lambda^*_{[\mathcal{S}_1]}(v) \sqsubseteq \lambda^*_{[\mathcal{S}_2]}(w) \mid \langle v, w\rangle \in \phi, \phi \in \Phi\}, \tag{29}$$

where $v \in \mathcal{S}_1$ and $w \in \mathcal{S}_2$.

6. Implementation and Working Example

In this section we present the implementation of our method for explaining non-entailments of semantic matching in $\mathcal{EL}_\bot$, described in Section 5. We illustrate the implementation using a working example and present results from simulated scenarios and experiments on realistic ontologies.

6.1. Implementation

Currently, the algorithm is implemented in Java using the OWL API [55]. The classification and entailment checking is performed by the Pellet reasoner [56].

The input to the algorithm is a non-entailment of a certain semantic match. More specifically, the algorithm receives some background knowledge ($\mathcal{T}$), matching concept descriptions (C and D), and a type of a match (exact ($\equiv$), plugin ($\sqsubseteq$), or subsume $\sqsupseteq$), which is an abduction problem of the form defined in Definition 6. The proposed method is implemented in four main steps:

Step 1. Read input parameters for the abduction problem (background knowledge $\mathcal{T}$, concept descriptions C and D, and type of match $\sqsubseteq$).
Step 2. Get the description trees T_C and T_D for the inputted concept descriptions C and D w.r.t. Definition 7.
Step 3. Compute the set Φ of $\rho_{\subseteq}$-subtree isomorphisms between T_C and T_D obtained in **Step 2** w.r.t. proposed methodology in Section 5.
Step 4. Construct the set of hypotheses explaining the non-entailed semantic match w.r.t. Eq. (29) from the set Φ obtained in **Step 3**.

We will now present the implementation for Steps 2-4 of our method.

Step 2. In this step the respective description trees of the inputted concept descriptions are obtained. To do this, we first impose a standardization on inputted concept definitions.

Definition 17. *Let $\mathcal{T}$ be an $\mathcal{EL}_\bot$ TBox and let C be a concept description defined w.r.t. the signature of $\mathcal{T}$. Then, C is standardized if $C = \prod_{i=0}^{n} A_i \sqcap (\prod_{i=0}^{m} (\exists r_i.C_i))$ and for all $C_i, 0 \leqslant i \leqslant m$ in C, C_i is standardized.*

The algorithm that translates a concept description into a description tree is represented in Appendix **??** Alg. **??**. Essentially, it splits the expression into two parts: the conjunction of atomic concepts, and conjunction of role restrictions. It creates a node and adds the set of concept conjuncts as a label of that node. Next, it is iterated over each role restriction and the children of the current node are constructed. Each edge is labeled w.r.t. the role names. Following this, the concepts restricted by the roles are added next in queue and the same is done for them. This follows the inductive definition of description trees presented in Section 5. Each concept description is written in OWL Manchester Syntax.

Step 3. This step contains the main algorithm for computing subtree isomorphisms between description trees. It is the main implementation of our methodology. Here are the steps of the algorithm for computing $\rho_{\subseteq}$-subtree isomorphisms between description trees:

Step 3.1 Read inputted description trees of concepts.
Step 3.2 Initialize the set of $\rho_{\subseteq}$-subtree isomorphisms Φ.
Step 3.3 Initialize the solutions tree and its root by mapping the roots of the description trees.
Step 3.4 Add the root node of the solutions tree to the queue.

Step 3.5 Obtain the next node from the solutions tree in the queue.
Step 3.6 For each pair of mapped vertices in the current node find all children in the respective description trees and partition them w.r.t. the equivalence relation in Theorem 2.
Step 3.7 Construct the relation $\mu(v,w)$ using Eq. (19) for the pair of vertices v and w.
Step 3.8 Construct $I(v,w)$ using Eq. (20).
Step 3.9 Construct and save $\mathcal{M}(v,w)$ for the current pair of vertices using Eq. (21).
Step 3.10 If there are no more pairs of vertices to evaluate go to the next step, otherwise go to **Step 3.6**.
Step 3.11 For each $\mathcal{M}(v,w)$ find all combinations of mapped vertices and construct a new node in the solutions tree and an edge between the current node and newly constructed node.
Step 3.12 If the queue is empty continue, otherwise go to **Step 3.5**.
Step 3.13 Find all complete paths in the solutions tree and construct Φ using Eq. (24).

To illustrate how we compute $\rho_{\sqsubseteq}$-subtree isomorphisms between descriptions trees and construct the solutions tree, consider the following example TBox related to medical imaging techniques:

$$\begin{aligned}\mathcal{T} = \{&\exists\text{produces.RadioWave} \sqcap \exists\text{produces.SoundWave} \sqsubseteq \bot, \text{SoundWave} \sqcap \text{RadioWave} \sqsubseteq \bot,\\ &\text{RadioWave} \sqsubseteq \text{Non-IonizingRadiation}, \text{IonizingRadiation} \sqcap \text{Non-IonizingRadiation} \sqsubseteq \bot\},\end{aligned} \tag{30}$$

with signature $\Sigma_{\mathcal{T}} = \langle \mathcal{N}_{\mathcal{C}}, \mathcal{N}_{\mathcal{R}} \rangle$, where:

$$\begin{aligned}N_{\mathcal{C}} &= \{\text{Imaging}, \text{Magnet}, \text{Material}, \text{RadioWave}, \text{X-Ray}, \text{X-RayComputedTomography}, \text{IonizingRadiation},\\ &\quad \text{Non-IonizingRadiation}, \text{X-RayTube}, \text{Rotation}, \text{Part}\}\\ N_{\mathcal{R}} &= \{\text{uses}, \text{performs}, \text{produces}, \text{involves}\}\end{aligned}$$

Let **CTScan**, and **X-RayScan** be two concepts defined w.r.t the signature $\Sigma_{\mathcal{T}}$, such that:

$$\begin{aligned}\textbf{CTScan} = \text{Scan} &\sqcap \exists\text{performs.X-RayComputedTomography}\\ &\sqcap \exists\text{uses.}(\text{X-RayTube} \sqcap \exists\text{involves.Rotation}) \sqcap \text{produces.X-Ray},\end{aligned} \tag{31}$$

$$\textbf{X-RayScan} = \text{Scan} \sqcap \exists\text{performs.Imaging} \sqcap \exists\text{uses.Part} \sqcap \exists\text{produces.IonizingRadiation}. \tag{32}$$

In this working example we have an abduction problem $\langle \mathcal{T}, \text{match}_{\sqsubseteq}(\text{CTScan}, \text{X-RayScan}) \rangle$. It can be easily verified that $\mathcal{T} \not\models \text{CTScan} \sqcap \text{X-RayScan} \equiv \bot$ and $\mathcal{T} \not\models \text{CTScan} \sqsubseteq \text{X-RayScan}$.

The description trees of the concepts **CTScan** and **X-RayScan**, as well as the solutions tree are shown in Figure 5.

By definition, the roots of description trees need to be mapped in order to obtain an isomorphism. Thus, we construct the root of the solutions tree carrying the mapping $\langle v_0, w_0 \rangle$ and denote the root of the solutions tree as ϕ_0. Next, the children of the roots v_0 and w_0 are partitioned w.r.t. the edge labels, from which we get:

- $N(v_0)_{/\sim} = \{[v_1], [v_2], [v_4]\}$, and
- $N(w_0)_{/\sim} = \{[w_1], [w_2], [w_3]\}$.

This partitioning is done according to the equivalence relation defined in Theorem 2. In the example, the equivalence classes contain the following vertices in T_1:

- $[v_1] = \{v_1\}, [v_2] = \{v_2\}, [v_4] = \{v_4\}$,

and in T_2:

- $[w_1] = \{w_1\}, [w_2] = \{w_2\}, [w_3] = \{w_3\}$.

Further, the relation from Eq. (19) is defined for v_0 and w_0 using the sets of partitions $N(v_0)_{/\sim}$ and $N(w_0)_{/\sim}$, from which we get:

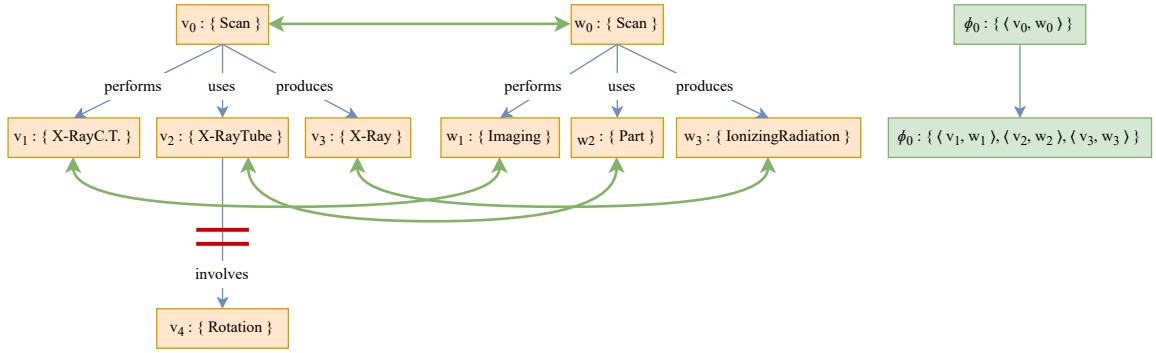

Fig. 5. A Solutions Tree containing the subtree isomorphism between the concepts CTScan and X-RayScan.

– $\mu(v_0, w_0) = \{\langle [v_1], [w_1] \rangle, \langle [v_2], [w_2] \rangle, \langle [v_4], [w_3] \rangle\}$.

The relation $\mu(v_0, w_0)$ pairs the corresponding equivalence classes of the children of v_0 and w_0. An equivalence class obtained from a specific edge label in one tree is paired with an equivalence class obtained from an equal edge label in the other tree. Then, the relation $\mu(v_0, w_0)$ is used to map the corresponding equivalence classes.

We are interested in all isomorphic mappings. Thus, to find all subtree isomorphism, we need to search for injective maps between pairs of equivalence classes (Eq. (20)). For the example, the set of injective mappings between equivalence classes is:

– $I(v_0, w_0) = \{\{\langle v_1, w_1 \rangle\}, \{\langle v_2, w_2 \rangle\}, \{\langle v_4, w_3 \rangle\}\}$,

Finally, all unique combinations of injections are taken using Eq. (21):

– $\mathcal{M}(v_0, w_0) = \{\{\langle v_1, w_1 \rangle, \langle v_2, w_2 \rangle, \langle v_4, w_3 \rangle\}\}$.

A new node ϕ_1 in the solutions tree is created that carries the mapped nodes from $\mathcal{M}(v_0, w_0)$, such that $\lambda_{\mathcal{V}_\Phi}(\phi_1) = \{\langle v_1, w_1 \rangle, \langle v_2, w_2 \rangle, \langle v_4, w_3 \rangle\}$. If there were more than one set of mapped vertices between the trees, then for each set a new node is constructed and added in the solutions tree. To complete this iteration, the node ϕ_1 from the solutions tree is added next in the queue. The algorithm goes over each pair of mapped vertices from the queued solutions tree node and performs the same steps. This procedure ensures that in each step only vertices that preserve the structure between the description trees will be mapped, while the ones that do not preserve the structure will not be mapped to any vertices and will induce a subtree in their respective description tree.

In the next step the mapped vertices in the pairs $\langle v_1, w_1 \rangle$, $\langle v_2, w_2 \rangle$, and $\langle v_4, w_3 \rangle$ are to be evaluated. As before, their children are partitioned and we get:

– $N(v_1)_{/\sim} = \{\}$ and $N(w_1)_{/\sim} = \{\}$,
– $N(v_2)_{/\sim} = \{[v3]\}$ and $N(w_2)_{/\sim} = \{\}$, and
– $N(v_4)_{/\sim} = \{\}$ and $N(w_3)_{/\sim} = \{\}$,

from which the mapping relations are formulated w.r.t. Eq. (19):

– $\mu(v_1, w_1) = \{\}$,
– $\mu(v_2, w_2) = \{\}$,
– $\mu(v_4, w_3) = \{\}$.

We can see that in this iteration there exist no nodes from either description tree that could be mapped in order to obtain an isomorphism. All nodes that could not be mapped in this iteration, because they are tails of edges that do not have corresponding labels and mapping them would not provide an isomorphism, are omitted from the final abductive solution. Thus, the method stops this iteration and goes onto the next. Since no new nodes are added in

the queue to be evaluated, the method returns the solutions tree with all possible $\rho_{\subseteq}$-subtree isomorphisms found in each unique path from the root to leaves in the tree.

From the solutions tree on Figure 5 we can see that the set of all complete paths is $\Pi^0 = \{\langle \phi_0, \phi_1 \rangle\}$. In this case, we only have one $\rho_{\subseteq}$-subtree isomorphism. In cases where there are more than one solution, we run a simple breadth-first search (BFS) to obtain the set of all complete paths in the solutions tree. From there, we construct the set of all weak $\rho_{\subseteq}$-subtree isomorphisms, Φ, which for our running example:

– $\Phi = \{\{\langle v_0, w_0 \rangle, \langle v_1, w_1 \rangle, \langle v_2, w_2 \rangle, \langle v_4, w_3 \rangle\}\}$,

where Φ contains the weak subtree isomorphism between the description trees T_{CTScan} and $T_{\text{X-RayScan}}$.

The isomorphisms are built from top to bottom and the equivalence relation that partitions the nodes w.r.t. the edge labels, as well as the relation $\mu(v, w)$ that partitions the equivalence classes w.r.t. corresponding edge labels in both trees, adhere to the definition of weak isomorphisms and do not allow for mapping of vertices that do not preserve the structure between description trees. Moreover, those edges and vertices that do not preserve the structure between the description trees, and consequently between the descriptions of the matching concepts, are omitted from the final set of mapped vertices, thus inducing subtrees of their respective trees. The complete algorithm for computing weak isomorphisms between $\rho_{\subseteq}$-subtrees of description trees is shown in Algorithm 1.

Step 4. The final step of the main code takes as an input a set of weak $\rho_{\subseteq}$-subtree isomorphisms between description trees. It then iterates over all weak $\rho_{\subseteq}$-subtree isomorphisms, and for each one it construct the missing relation between concepts and/or role restrictions w.r.t. Eq. (29), thus extending the set of weak isomorphisms to $\mathcal{T} \cup \mathcal{H}$-isomorphisms w.r.t. Definitions 9 and 12. This is done for each obtained weak $\rho_{\subseteq}$-subtree isomorphism between the description trees. The steps of this algorithm are as follows:

Step 4.1 Read the set of weak $\rho_{\subseteq}$-subtree isomorphisms, Φ.
Step 4.2 For each $\phi \in \Phi$, update the labels of vertices using Eq. (28).
Step 4.3 Construct the hypothesis for each ϕ w.r.t. Definitions 9 and 12.

The algorithm for constructing the hypotheses from a set of weak $\rho_{\subseteq}$-subtree isomorphisms is shown in Algorithm 4 Appendix A.3. The algorithm traverses over each weak $\rho_{\subseteq}$-subtree isomorphism and constructs the hypothesis as in Eq. (29). All vertices that are not included in the isomorphic mappings, because they do not satisfy the conditions for a weak isomorphism, induce $\rho_{\subseteq}$-subtrees of their respective description trees. The labels of the vertices that are included in the weak $\rho_{\subseteq}$-subtree isomorphism are then updated with the role restrictions of the concepts in the labels of vertices that were omitted from the weak $\rho_{\subseteq}$-subtree isomorphism, according to Definition 12.

In our working example, we have one isomorphism, $\Phi = \{\langle v_0, w_0 \rangle, \langle v_1, w_1 \rangle, \langle v_2, w_2 \rangle, \langle v_4, w_3 \rangle\}$, from which we obtain the hypothesis:

– $\mathcal{H} = \{\text{X-RayC.T.} \sqsubseteq \text{Imaging}, \text{X-RayTube} \sqcap \exists \text{involves.Rotation} \sqsubseteq \text{Part}, \text{X-Ray} \sqsubseteq \text{IonizingRadiation}\}$,

In natural language this hypothesis states that *"X-Ray computed tomography is a type of imaging."*, *"X-Ray tube that rotates is a part."* and *"X-Ray is a type of ionizing radiation."*. In reality, an X-Ray computed tomography is a type of an X-Ray scan, because it is based on the same technique. In fact, an X-Ray computed tomography is a specific type of an X-Ray scan. The hypothesis clearly states what is missing from our background knowledge in order for the concepts **X-RayC.T.** and **X-RayScan** to be in a plugin match.

6.2. Experiments

Since there is no standardized benchmark for $\mathcal{EL}$ and $\mathcal{EL}_\bot$ abduction, we performed two types of experiments: *i)* synthetic experiments in which we constructed random description trees with arbitrary concepts and performed abduction using our method, with the goal of stressing the computational capabilities of the method, and *ii)* experiments on realistic ontologies from the bio-medical domain [3]. All experiments were conducted on a 64-bit operating system running on Windows 10, CPU Intel(R) Core(TM) i5-7400 CPU @ 3.00GHz, RAM 8.00 GB.

[3]The code used to run the simulations and solve the working example is available on a GitHub repository https://github.com/Gochev-Ivan/An-Abduction-based-Method-for-Explaining-Non-entailments-of-Semantic-Matching

Algorithm 1 subtreeIsomorphisms(T_C, T_D)

$\Phi = \{\}$
$\mathcal{V}_\Phi \leftarrow \{0 : [\langle v_0, w_0 \rangle]\}$
$\mathcal{E}_\Phi \leftarrow \{\}$
queue $\leftarrow [0]$
$i \leftarrow 1$
while |queue| $\neq 0$ **do**
  $n \leftarrow$ queue.$poll()$
  uniqueMappings $\leftarrow []$
  for all $\langle v, w \rangle \in \mathcal{V}_\Phi[n]$ **do**
    currentMappingsEQC $\leftarrow$ $getMappingsOfEquivalenceClasses(v, w, T_C, T_D)$
    uniqueMappings.add(currentMappingsEQC)
  end for
  for all $x \in getCartesianProduct$(uniqueMappings) **do**
    $\phi_n \leftarrow []$
    for all $y \in x$ **do**
      if $|y| = 0$ **then**
        continue
      end if
      $\phi_n.addAll(y)$
    end for
    if $|\phi_n| = 0$ **then**
      continue
    end if
    $\mathcal{V}_\Phi.put(i, \phi_n)$
    $\mathcal{E}_\Phi.add(\langle n, i \rangle)$
    queue.$add(i)$
    $i \leftarrow i + 1$
  end for
end while
$j \leftarrow 1$
for all $\pi \in allCompletePaths(\mathcal{V}_\Phi, \mathcal{E}_\Phi)$ **do**
  $\phi \leftarrow []$
  for all $node \in \pi$ **do**
    for all $\langle v, w \rangle \in \mathcal{V}_\Phi[node]$ **do**
      $\phi.add(\langle v, w \rangle)$
    end for
  end for
  $\Phi.put(j, \phi)$
  $j \leftarrow j + 1$
end for
return Φ

Simulations. For the synthetic experiments, concept descriptions were randomly generated with different number of concepts and roles, with allowed repetition of role restrictions on various concepts. This way we also cover extreme cases in which concepts are restricted by the same role multiple times, possibly leading to a large number of abductive solutions. In each simulation two random numbers were taken from a uniform distribution in a provided interval. We constructed three scenarios each with an interval of: **1)** $[1, 10]$, **2)** $[11, 20]$, and **3)** $[21, 30]$. For each scenario, in each simulation, a number from the provided interval was picked that represents the number of concepts in the current description (c) and a second number from the interval $[1, 10]$ was picked to represent the maximal

number of roles the concept description is allowed to have (r). Next, a loop was designed to restrict c number of concepts each with a 50% chance to be restricted by a role. The role was picked randomly from a uniform distribution of a set of roles with cardinality r, i.e. a maximal number of roles that were allowed for that concept. For example in scenario **1)** there could be anywhere from 1 to 10 concepts in the description and 1 to 10 possible roles to pick from and restrict that concept, each role given an equal chance to be picked. This provides the equal possibility for concepts to be restricted by the same role/s multiple times, so that the scalability of the method can be tested. If sequential iterations happen to restrict concepts, then nested expressions were formulated. Provided that a concept was picked not to be further restricted by a role, the concept in the following iteration would represent once again a top level conjunct in the description.

We then obtained the description trees for the randomly generated concept descriptions and performed abduction using our method. We report on the above mentioned three scenarios. For each scenario we ran 250 simulations with the given parameters. The non-entailment of interest was designed as $\mathcal{T} \not\models \text{match}_{\sqsubseteq}(C, D)$, where C and D are the randomly generated concept descriptions. For each solution we constructed a new ontology, such that $\mathcal{T} = \emptyset$ in which we added the atomic concepts and roles and the matching concept definitions from those trees, thus generating a signature for the TBox and a non-entailment to explain. We then added the hypotheses to each distinct ontology and checked whether $\mathcal{T} \cup \mathcal{H} \models \text{match}_{\sqsubseteq}(C, D)$, i.e. whether the non-entailment has been explained by the certain hypothesis.

Because our methodology focuses on finding abductive solutions that contain direct relations between concept descriptions and does not use information in the background knowledge to find connecting concepts, we opted to go for initially empty TBoxes in the synthetic experiments. In real scenarios, concepts can have multiple definitions in the background knowledge. For example, for some terminological knowledge $\mathcal{T}$, if we have a non-entailment $\mathcal{T} \not\models A \sqsubseteq \exists r.B$ we would obtain an abductive solution that contains $A \sqsubseteq \exists r.B$. However, if the TBox contains information such that $\mathcal{T} \models A \equiv \exists r.D$, then the outcome of the abduction could be different. If we take into account that the definition on the left-hand side expression, A can be extended to $\exists r.D$, we would then get a non-entailment of the form $\mathcal{T} \not\models \exists r.D \sqsubseteq \exists r.B$, for which the abductive process will yield $D \sqsubseteq B$, provided that the TBox does not already contain abducted information.

If we find that there are more than one definition of a matching concept, then we can perform our abduction process (activate our method) for all (or a chosen subset of) definitions w.r.t. the matching concepts. For example, if we have the non-entailment $\mathcal{T} \not\models A \sqsubseteq \exists r.B$ and the concept A has two definitions in $\mathcal{T}$, $A \equiv \exists r.D$ and $A \equiv C \sqcap \exists r.E$, then we can perform abduction on $\exists r.D \sqsubseteq \exists r.B$ and $C \sqcap \exists r.E \sqsubseteq \exists r.B$. In addition, if the concept A is subsumed by multiple descriptions in the background knowledge, we can construct a new definition of the concept A as a conjunction of all descriptions. The latter was done in the experiments on real-world ontology datasets.

Results We measured the mean, median, and maximal values of the cardinalities of vertex sets ($|V_1|, |V_2|$), the number of solutions (hypotheses) for the given abduction problem ($\#H$), the size of the solutions ($|H|$), the time needed to compute the subtree isomorphisms and checked whether the generated hypotheses explain the non-entailments. The simulations and results are shown in Table 1.

Table 1
Results from the synthetic experiments.

mean | median | max

	$\lvert\mathcal{V}_C\rvert$	$\lvert\mathcal{V}_D\rvert$	$\#\mathcal{H}$	$\lvert\mathcal{H}\rvert$	**t[s]**	$\mathcal{T} \cup \mathcal{H} \models \eta$ **[%]**
1	6.03 | 6.0 | 10	5.9 | 6.0 | 10	4.71 | 1.0 | 1680	1.77 | 2.0 | 7.0	0.002 | 0.0 | 1.156	100.00
2	15.55 | 16.0 | 20	15.47 | 16.0 | 20	10.34 | 2.0 | 7776	3.22 | 3.0 | 12.0	0.007 | 0.0 | 4.803	100.00
3	25.49 | 26.0 | 30	25.51 | 26.0 | 30	37.41 | 2.0 | 10368	4.56 | 4.0 | 17.0	0.04 | 0.001 | 11.565	100.00

Figure 6 represents the times needed to compute a certain number of solutions that an abduction problem has. On the x-axis are shown standardized values for the number of solutions (hypotheses - #H) for each abduction problem, and on the y-axis the time (in seconds) needed to compute all solutions. The results demonstrate that the method can practically and correctly compute hypotheses to explain $\mathcal{EL}$ non-entailments. For smaller concept descriptions in the first and second scenario, that produce smaller trees, the method computes the hypotheses for the given

abduction problem almost instantaneously. Even for larger concept descriptions that produce larger trees (Scenario 3) with maximum of 30 vertices the method is quite versatile. On Figure 7 we present each of the scenarios and the actual number of solutions for the abduction problems and the times needed to compute those. We can see that there is some deviation in the results that show distinct cases where the method needs more time to compute subtree isomorphisms, which is due to the frequency of repetitive (identical) edge labels in description trees.

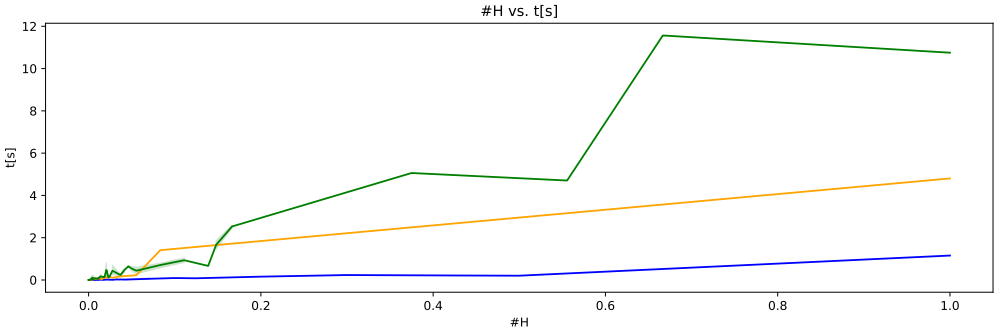

Fig. 6. Simulation results: (Standardized) Number of Hypotheses (#H) vs time (t[s]) for Scenarios **1**), **2**), and **3**).

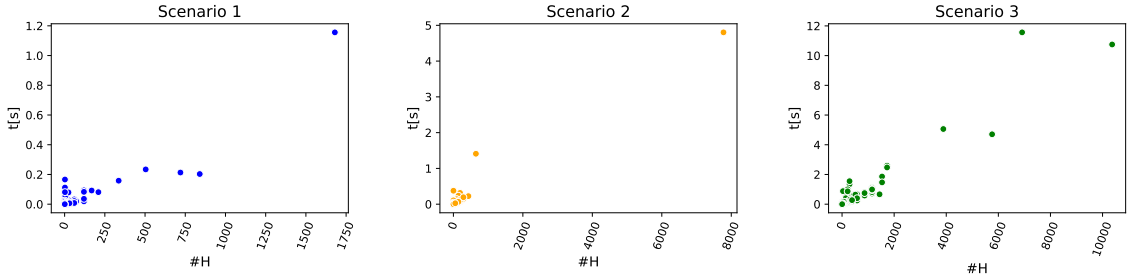

Fig. 7. Simulation results: (Actual) Number of Hypotheses (#H) vs time (t[s]) for Scenarios **1**, **2**, and **3**.

Additionally, we obtained the average branching factors of the description trees and the sets of edge labels of description trees, which we define as $\Sigma_R(\mathcal{E}) = \{\lambda_{\mathcal{E}}(e) \mid e \in \mathcal{E}\}$. We then calculated the ratio between the average branching factors and the Jaccard index of the sets of edge labels defined as:

$$J(\Sigma_R(\mathcal{E}_1), \Sigma_R(\mathcal{E}_2)) = \frac{|\Sigma_R(\mathcal{E}_1) \cap \Sigma_R(\mathcal{E}_2)|}{|\Sigma_R(\mathcal{E}_1) \cup \Sigma_R(\mathcal{E}_2)|} \tag{33}$$

This was done with the purpose of observing how the structure of description trees and the number of overlapping edge labels influence the number of solutions of the abduction problems.

Figure 8 presents the positive correlation between the Jaccard indexes of sets of edge labels of descriptions trees and the number of solutions for the abduction problems in those cases. Essentially, if there is a higher similarity of edge labels in description trees, then there will be larger equivalence classes and more combinations of mapped nodes that represent abductive solutions. This is due to our approach of partitioning the vertices on specific depths in the trees based on the labels of edges. In order to obtain isomorphic mappings we need to map edges that have same edge labels (Definition 8 condition 2). Moreover, we obtain mappings between vertices that belong in partitions coming from same edge labels. Consequently, if there is a lower similarity of the sets of edge labels, or even no

common edge labels, then there will be a lower number (or even none) partitions and vertices to map. Thus, the method omits those mappings between vertices that do not represent an isomorphism. These are the cases in which the method almost instantaneously computes subtree isomorphisms. On the other hand, higher similarity of edge labels in the trees may lead to a large number of hypotheses.

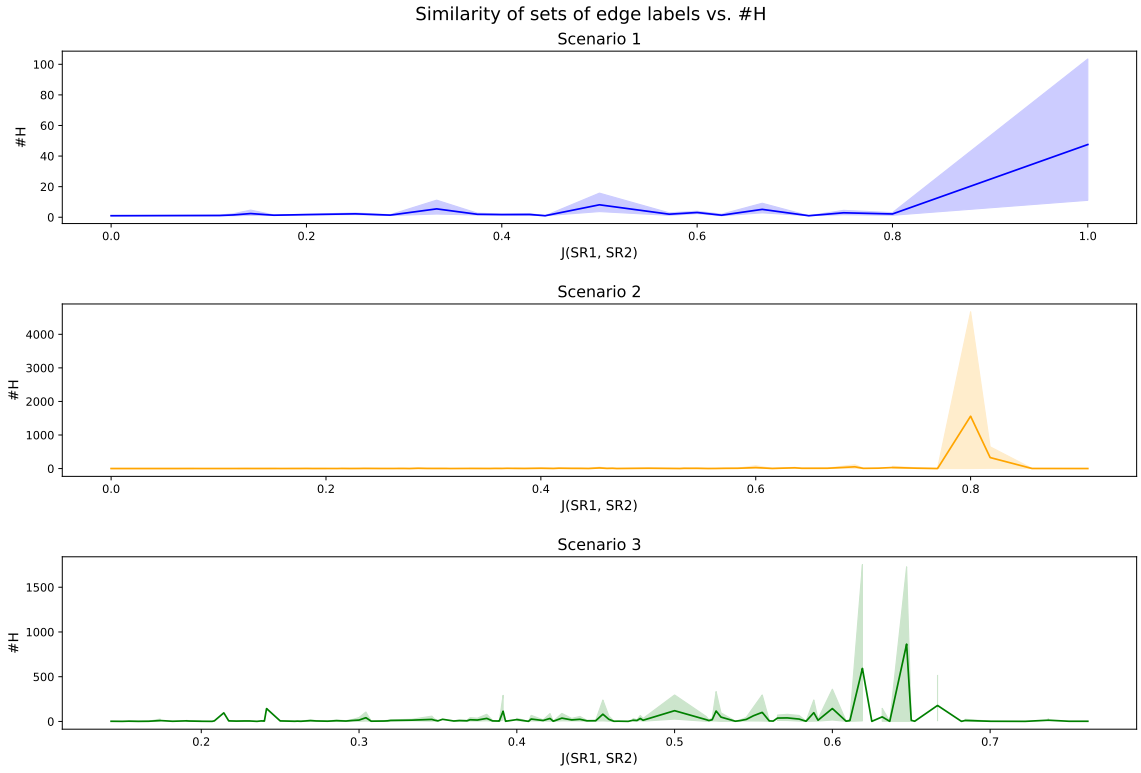

Fig. 8. Simulation results: Jaccard indexes of sets of edge labels of trees vs (Actual) Number of Hypotheses (#H) for Scenarios **1**, **2**, and **3**.

To add to the explanation for the similarity of edge labels, the last correlation that was observed is shown in Figure 9, which shows the ratios of branching factors and the number of solutions for abduction problems for the three scenarios. Since we are dealing with isomorphisms, the ratio of the average branching factors is calculated as the lower value divided by the larger value. A lower ratio of the branching factors between trees could potentially induce a higher difference of the sizes of equivalence classes. Thus, this will yield more combinations of isomorphic mappings, and, consequently, more solutions to abduction problems. This occurred mostly around a value for the ratio of the average branching factors of ≈ 1.

The average branching factor represents the average number of children at each node. If the average branching factor of a tree is high, then there are more nodes to partition into equivalence classes. However, this does not necessarily indicate larger equivalence classes and more solutions. If there is a high average branching factor of a tree *and* only a very few unique edge labels (roles) that partition those nodes, then there will be fewer equivalence classes that will contain a high number of nodes. Thus, if two trees have a ratio of their average branching factors ≈ 1, and there are very few roles introduced in the edge labels in both trees, then the sizes of the equivalence classes in $I(v, w)$ (Eq. 20) will increase drastically. Specifically, the number of mappings between equivalence classes is $\frac{||[x]||!}{(||[x]||-||[y]||)!}$. For example, for two trees T_1 and T_2, if a node $v \in T_1$ has 10 children and a node $w \in T_2$ has 10 children, and all children of v and w in T_1 and T_2 are tails of edges with the same label, then there will be one equivalence class in T_1 with $||[x]|| = 10$ and one equivalence class in T_2 with $||[y]|| = 10$. Hence, the number of

mappings between the equivalence classes will be 10!. Thus, a limitation of our method is that it may not be able to practically compute all hypotheses in specific cases where the trees of matching concepts have a ratio of average branching factors ≈ 1 and high value for the Jaccard index. In this cases a timeout should be introduced.

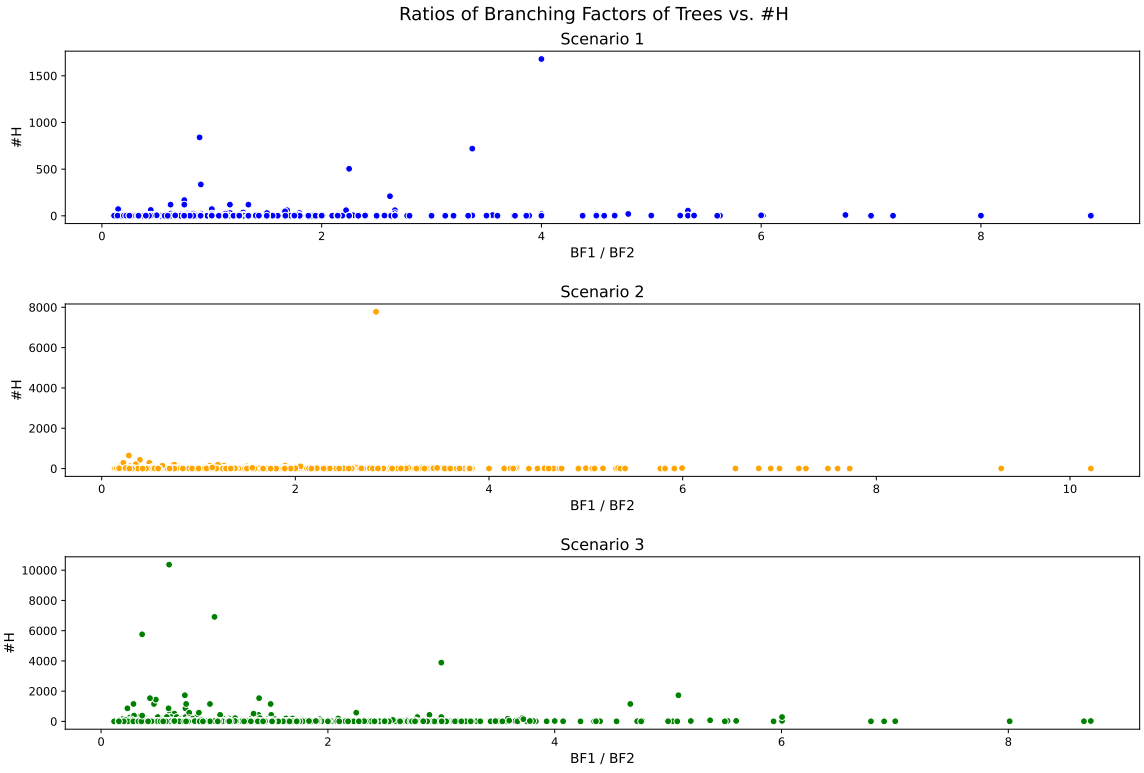

Fig. 9. Simulation results: Ratios of Branching Factors of Trees vs (Actual) Number of Hypotheses (#H) for Scenarios **1**, **2**, and **3**.

6.3. Experiments on real-world ontology datasets

In addition to performing an experimental evaluation on randomly generated concept descriptions (synthetic experiments), to stress the computational capabilities of the method, we performed an experimental evaluation on realistic ontologies from the bio-medical domain (2017 snapshot of BioPortal [57]) and the LUBM ontology [60].

Each ontology in the corpus is restricted to its $\mathcal{EL}$ / $\mathcal{EL}_\bot$ fragment of the TBox, such that axioms containing constructs other than those supported by $\mathcal{EL}$ / $\mathcal{EL}_\bot$ ontologies were removed. No additional transformations or restrictions were made to the ontologies, i.e. the remaining axioms were left in their original form. Table 2 shows the results from the experiments performed on real ontology datasets. We report on the number of abduction problems that were generated (#AP) of the form $\langle \mathcal{T}, \text{match}_{\sqsubseteq}(C, D)\rangle$, i.e. two random concepts were picked from the ontology and matched (but with the disadvantage that the picked concept may not necessarily be related in the background knowledge), percentage of time the method terminated (Complete - C), successfully found non-empty solutions when the method terminated (Success - S), and whether the solution was trivial (Trivial Solution - TS). We measured the mean, median, and maximal values for the number of solutions ($\#\mathcal{H}$) that were generated, the size of the solutions ($|\mathcal{H}|$), the number of concepts that were included in solutions ($\#C$), the number of role restrictions that were included in solutions ($\#R$), and the time needed to compute the solutions ($t[s]$).

Table 2
Results from the experiments on real ontology datasets.

	#AP	C | S | TS [%]	mean | median | max $\#\mathcal{H}$	$\lvert\mathcal{H}\rvert$	$\#C$	$\#R$	t[s]
BP	3086	99.42 | 99.42 | 89.63	2.29 | 2.0 | 380.0	1.08 | 1.0 | 5.62	4.27 | 4.0 | 82.0	0.62 | 0.0 | 382.0	0.0 | 0.0 | 0.067
LUBM	250	98.40 | 98.40 | 98.40	1.95 | 2.0 | 2.0	1.0 | 1.0 | 1.0	3.84 | 4.0 | 4.0	0.06 | 0.0 | 1.0	0.0 | 0.0 | 0.013

Out of the 438 ontologies in the BioPortal dataset, 316 were successfully loaded and reasoned with. After restricting each ontology to its $\mathcal{EL}$ / $\mathcal{EL}_\bot$ fragment, we attempted for each loaded ontology to create 10 abduction problems by matching two random concepts from the background knowledge. This would give in total 3160 problems to solve, but the attempt failed 74 times out of the entire dataset (3160). The reasons for the failed attempt were loading null objects because the loaded ontology did not contain any concepts or failed to load the signature of the ontology. We then took direct definitions of the matching concepts that already exist in the background knowledge and constructed a conjunction, thus formulating the final descriptions in the abduction problem. The same was done for the LUBM ontology. Since the size of the LUBM ontology is small (contains 43 concepts and 32 roles), we created 250 abduction problems for it. We then ran our method for the constructed abduction problems.

For both data sets we can see that the method successfully completes and finds solutions for the generated abduction problems. Because the concept matching scenario has the disadvantage of picking unrelated concepts that may have definitions containing completely different roles, the trivial solution was computed as the only solution in most cases in the BioPortal dataset and all cases in the LUBM ontology when the method successfully terminated. This is the reason why in most cases the method computes the abductions almost instantaneously. For the other cases, the method showed practical and successfully generated solutions containing abducted concepts and role restrictions.

In comparison to the similar method in [15], for the BioPortal data, our method managed to compute abductive solutions faster. When it comes to the sizes of abductive solutions w.r.t. the number of concepts and roles included, our method managed to obtain more concise abductive solutions. In addition, our method managed to obtain a slightly higher success rate, which is a direct consequence of the possibility to abduct role restrictions. However, the current state of our method is obtaining direct connections between concepts in a non-entailment, whereas [15] computes abductive solutions between intermediate connecting concepts, which needs additional time. Hence, as part of our future work is integrating our method with a technique to search for potentially connecting concepts within the background knowledge and compute abductive solutions between those concepts and perform a more thorough comparison.

Compared to [16], we only managed to obtain the LUBM data set. For this data set, our method provided abductive solutions more quickly. Nonetheless, the runtime in [16] is heavily dependent on computing patterns to use and obtain abductions, but this provides the possibility to generate more solutions. Still, we generate fewer and more concise explanations. However, a variety of patterns could in some cases present more explanations that would adhere to the users' preferences.

7. Discussion

Although our method is focused on explaining non-entailments of semantic matching, our main contribution is the extension of the state of the art method for abduction in ontologies, by generalizing abduction to axioms consisting of role restrictions too, and not only of atomic concepts.

The utilization of homomorphisms between description trees to solve abduction problems in $\mathcal{EL}$, that provide minimal connections between concepts was initially presented in [15]. The abduction consists of axioms that carry atomic concepts only. Even though our method finds direct relations between concepts in semantic matching, instead of finding connecting concepts in some background knowledge, the abduction of role restrictions that we introduce can enhance the method in [15] by finding connecting concepts that would otherwise be excluded if the abductive process is restricted to concepts only. We achieved this by inducing specific subtrees termed $\rho_\subseteq$-subtrees, which retain the original root and help identify non-isomorphic parts of description trees. The final solution contained the isomorphic parts, which enabled abduction of concepts, whereas the non-isomorphic parts of trees were used to

update existing labels, which enabled abduction of role restrictions. The experiments on realistic ontology datasets also showed that the method can successfully abduct concepts and role restrictions. Thus, state-of-the-art methods that use similar approaches could benefit by introducing abduction of role restrictions, instead of constraining the set of abducibles to concepts only. In addition to our method preserving the structure of concept definitions it does not unnecessarily omit parts relevant to the abduction.

The problem of abduction of concepts as well as role restrictions is tackled in [16]. This is done by constraining the abductive process by a set of predefined patterns that can be abducted. Apart from specializing our method for semantic matching, we do not constrain the form of expressions included in the abducted axioms; instead, our method is constrained only by the signature of the background knowledge. This will assure not to overlook certain patterns of nested expressions. Although we compute abductive solutions in a more timely manner and generate fewer and more concise solutions, the generation of patterns based on justifications that are used for formulating abductive solutions offer a large variety of formats of abductive solutions that can be used as explanations and for ontology debugging.

In [45, 47, 54] abduction methods for semantic matching in more expressive DLs are presented. In comparison, instead of reducing matching concept descriptions or introducing new concepts to reach a positive semantic match, our method searches for direct relations between concepts and role restrictions in matching concepts, thus preserving the original definitions of matching concepts.

Subtree isomorphisms have also been previously used with ontologies. In [58], subtree isomorphisms have been used to determine term equivalence to rewrite finite formal languages. The approach can significantly reduce the size of ontologies by capturing repeated expressions. Similarly, we use node and edge labeled trees, but we capture semantic relations through subtree isomorphisms instead, rather than terms. In addition, they propose equivalence classes of macros, that capture ontology terms that are equivalent, whereas we partition nodes of description trees into equivalence classes that can capture semantic relations. Furthermore, our subtree isomorphism definition is a variant of their definition, in which they search for subtrees in one tree that are contained in another entire tree. Our methodology focuses on preserving the semantic structure of concept definitions and thus introduces a condition that the roots of the trees of the original concept definitions must be mapped.

In [59] subtree isomorphisms are used to match video patterns represented by ontologies. The approach uses a naive search to obtain maximal subtree isomorphisms, whereas we employ *i)* trees that are additionally edge-labeled, and *ii)* a top-down approach using the edge labels to partition nodes and accumulate mappings that are isomorphic. Still, if we have description trees that contain only a unique edge label, then our method is generalized and can be said that solves the problem of computing subtree isomorphisms on unlabeled rooted trees. This can drastically increase the number of solutions in our case. A solution to this, which we currently work on, would be to introduce heuristics based on the complete paths in trees that would compute approximative solutions, but would decrease the method's runtime.

On the more general note, the authors in [61] offer a method for finding the largest possible subtree of a tree that is isomorphic to some other tree. There are several differences between the method in [61] and our method. First, the problem in [61] is defined as a decision problem, i.e. the algorithm answers the question whether there exists a subtree of one tree that is isomorphic to another given tree. It can also be transformed to a search problem to find a solution that satisfies this condition. However, our problem is defined as finding subtrees of both trees that are isomorphic, as opposed to finding if a subtree of one tree is isomorphic to an entire other tree. In addition, we look to compute all solutions, rather than decide whether there is a subtree isomorphism or find a solution that satisfies that condition. We compute all solutions in order to find all potential explanations for a non-entailment. However, a limitation of this is that the problem can exhibit a very high complexity. Next, we present a top-down approach that works with labeled directed rooted tree, whereas the method in [61] is bottom-up and focuses on unlabeled and not directed rooted trees. Finally, a bipartite graph is constructed from the children of nodes, for which the matching number (or a maximal matching) is computed, which will ultimately lead to deciding whether the answer to the problem is positive, or to find a solution that satisfies that condition. This is similar to our way of partitioning the nodes w.r.t. edge labels in description trees in equivalence classes and finding the bijective mappings between those classes. We opt to compute all bijections between equivalence classes, which could correspond to finding perfect matchings in bipartite graphs. This can potentially be implemented to $\mathcal{EL}$ description trees and the

search for isomorphic subtrees of two description trees, which could enhance the search for abductive solutions and produce them faster.

On the other hand, [62] offers a different approach for identifying subtree isomorphisms for unlabeled binary and ternary trees. Given two trees, they reduce the problem of finding a subtree of the one tree isomorphic to the other tree to the Orthogonal Vectors (OV) problem. The inputs to the OV problem are two lists of N vectors in $\{0,1\}^D$ and the output is positive if and only if there is a pair of vectors, one from each list, that are orthogonal [62]. This differs significantly to the method in this paper, in which we present a top-down approach to compute subtree isomorphisms between two trees by partitioning the vertices w.r.t. edge labels.

7.1. Limitations

One limitation of our method is scalability with large concept definitions. The method computes abductive solutions by partitioning the nodes in graphical representations of concepts, based on the roles that restrict nested concept expressions. It then matches concepts that are restricted by same roles in both concept definitions. If all role restrictions are done by the same role, i.e. we have a high repetition of role/s restrictions concepts, then the number of solutions can increase by a factor of $\frac{n!}{(n-m)!}$, where n represents the number of concepts restricted by a role in the first definition, and m represents the number of concepts restricted by the same role in the second definition. This depends heavily on how the background knowledge is built and how the matching concepts are defined. Therefore, a rich terminological knowledge that accustoms a variety of concepts and roles to introduce in definitions, would help improve the scalability of our method.

In [15], the abduction method via homomorphisms adheres to a connection-minimal criteria, which essentially connects two concept C and D via intermediate concepts. Thus, the search for homomorphisms is on (potentially connecting) concepts within the TBox, that connect two concepts in a non-entailment, i.e. $\mathcal{T} \not\models C \sqsubseteq D$, find C' and D', such that $T \models C \sqsubseteq C', \mathcal{T} \models D' \sqsubseteq D$, and the homomorphism is searched between the description trees of C' and D'. This approach provides the minimal connections between C and D. For the case of semantic matching in this paper, which is essentially brought down to subsumption axioms, we are looking for a *direct* connection between concepts C and D that have certain definitions w.r.t. the signature of the background knowledge. To this end, subtree isomorphisms are computed between the description trees of the matching concepts C and D, and not between the description trees of potentially connecting concepts C' and D'. The characteristic of the concept matching in this paper is having a direct connection between concepts, which we opt to find subtree isomorphisms that characterize those direct connections and not find intermediate concepts (and search subtree isomorphisms between those intermediate concepts) to connect two concepts in a non-entailed semantic match.

Although our method is specialized to find subtree isomorphisms and semantic layers explicitly between input concept definitions and it does not consider within the background knowledge alternative definitions that a matching concept may have, an extension can easily be integrated with our method. Currently, it only uses the background knowledge to verify that a solution to an abduction problem does not produce unsatisfiable concepts. However, an extension can be easily made to this. An initial solution would be to find all possible definitions for matching concepts w.r.t. some background knowledge and to perform abduction between all pairs of those definitions. Another possibility, as seen in our experimental setup, is to combine all definitions of a concept in a match into one single one. In addition, the search for alternative definitions of concepts can be enhanced by the existing semantics of the background knowledge. This would also render our method useful in other scenarios, e.g. more complex systems based on semantic matching and even ontology debugging and repair, and ontology completion. To this end, our ongoing work consists of two main things: *1)* formulating a heuristics based on the complete paths in description trees to assist our method in finding maximal subtree isomorphisms and abductive solutions w.r.t. our minimality criteria, and *2)* integrating our method with a search technique to find potentially related concepts within the background knowledge, for which we can find hypotheses and ultimately connect matching concepts.

8. Conclusion

We presented a method for explaining non-entailments of semantic matching in $\mathcal{EL}_\bot$ ontologies, by computing subtree isomorphisms between graphical representations of concept descriptions, i.e. description trees. To compute

isomorphic mappings, we started by partitioning the sets of neighboring nodes based on edge labels. The partitioned nodes from the respective trees were then mapped w.r.t. the labels. We formalized our approach and tested our method. The results showed that the method provides correct solutions within a reasonable time frame. We also discussed how our method could be used to enhance related methods, to achieve better results.

There are several ways to improve our method. First, the complexity of our method rises with the size of concept descriptions, therefore the size of description trees. Moreover, a combinatorial explosion occurs when there is higher repetition of roles in concept descriptions. A more thorough complexity analysis needs to be carried out. To improve the search for isomorphic mappings, we want to introduce heuristics based on the level of information concepts and roles carry in concept descriptions w.r.t. the background knowledge. This would return the hypotheses that carry the most information and exclude those that are similar or carry less information. In addition, we are researching the possibility of computing isomorphisms from paths in description trees, since paths from the root to a leaf node in rooted trees are unique, and preserved up to isomorphism, they could limit the search space even further. Finally, we want to look into the possibility of extending our method to more expressive description logics.

References

[1] Tiddi, I., 2020. Foundations of explainable knowledge-enabled systems. Knowl. Graph. eXplainable Artif. Intell.: Found. Appl. Challenges, 47, p.23.

[2] S. El-Sappagh, F. Franda, F. Ali, and K.-S. Kwak. Snomed ct standard ontology based on the ontology for general medical science. BMC medical informatics and decision making, 18:1–19, 2018.

[3] K. Degtyarenko, P. De Matos, M. Ennis, J. Hastings, M. Zbinden, A. McNaught, R. Alcántara, M. Darsow, M. Guedj, and M. Ashburner. Chebi: a database and ontology for chemical entities of biological interest. Nucleic acids research, 36(suppl_1):D344–D350, 2007.

[4] J. Liu, Y. Wang, J. Morris, and H. Kristiansen. Development of ontology for the anisotropic conductive adhesive interconnect technology in electronics applications. In Proceedings. International Symposium on Advanced Packaging Materials: Processes, Properties and Interfaces, 2005., pages 193–208. IEEE, 2005.

[5] G. Santos, F. Silva, B. Teixeira, Z. Vale, and T. Pinto. Power systems simulation using ontologies to enable the interoperability of multi-agent systems. In 2018 Power Systems Computation Conference (PSCC), pages 1–7. IEEE, 2018.

[6] A. Wiesner, A. Saxena, and W. Marquardt. An ontology-based environment for effective collaborative and concurrent process engineering. In 2010 IEEE International Conference on Industrial Engineering and Engineering Management, pages 2518–2522. IEEE, 2010.

[7] M. Sorli, I. Mendikoa, J. Pérez, A. Soares, L. Urosevic, D. Stokic, J. Moreira, and H. Corvacho. Knowledge-based collaboration in construction industry. In 2006 IEEE International Technology Management Conference (ICE), pages 1–8. IEEE, 2006.

[8] D. Beimel and S. Albagli-Kim. Enhancing medical decision making: A semantic technology-based framework for efficient diagnosis inference. Mathematics, 12(4):502, 2024.

[9] D. Arena, D. Kiritsis, C. Ziogou, and S. Voutetakis. Semantics-driven knowledge representation for decision support and status awareness at process plant floors. In 2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC), pages 902–908. IEEE, 2017.

[10] Kalyanpur A., Parsia B., Horridge M., Sirin E. (2007) Finding All Justifications of OWL DL Entailments. In: Aberer K. et al. (eds) The Semantic Web. ISWC 2007, ASWC 2007. Lecture Notes in Computer Science, vol 4825. Springer, Berlin, Heidelberg

[11] Alrabbaa, C., Borgwardt, S., Koopmann, P. and Kovtunova, A., 2022, December. Explaining ontology-mediated query answers using proofs over universal models. In Rules and Reasoning: 6th International Joint Conference on Rules and Reasoning, RuleML+ RR 2022, Berlin, Germany, September 26–28, 2022, Proceedings (pp. 167-182). Cham: Springer International Publishing.

[12] Alrabbaa, C., Borgwardt, S., Koopmann, P. and Kovtunova, A., 2022. Finding good proofs for answers to conjunctive queries mediated by lightweight ontologies. In DL Workshop.

[13] Alrabbaa, C., Baader, F., Borgwardt, S., Koopmann, P. and Kovtunova, A., 2021, July. Finding Good Proofs for Description Logic Entailments using Recursive Quality Measures. In CADE (Vol. 28, pp. 291-308).

[14] Alrabbaa, C., Baader, F., Borgwardt, S., Koopmann, P. and Kovtunova, A., 2020. Finding small proofs for description logic entailments: Theory and practice. In 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR23 2020 (pp. 32-67). EasyChair.

[15] Haifani, F., Koopmann, P., Tourret, S. and Weidenbach, C., 2022. Connection minimal Abduction in EL via Translation to FOL. Proc. IJCAR.

[16] J. Du, H. Wan, and H. Ma. Practical tbox abduction based on justification patterns. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 31, 2017.

[17] Wei-Kleiner, F., Dragisic, Z. and Lambrix, P., 2014, June. Abduction framework for repairing incomplete EL ontologies: Complexity results and algorithms. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 28, No. 1).

[18] L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In Proceedings of the 12th international conference on World Wide Web, pages 331–339, 2003.

[19] T. Di Noia, E. Di Sciascio, F. Donini, and M. Mongiello. Semantic matchmaking in a P-2-P electronic marketplace. In Proc. Symposium on Applied Computing (SAC '03), pages 582–586. ACM, 2003.
[20] S. Grimm. Intersection-based matchmaking for semantic web service discovery. In Second International Conference on Internet and Web Applications and Services (ICIW'07), pages 14–14. IEEE, 2007.
[21] M. Klusch. Semantic web service description. In CASCOM: intelligent service coordination in the semantic web, pages 31–57. Springer, 2008.
[22] U. Keller, R. Lara, H. Lausen, and D. Fensel. Semantic web service discovery in the wsmo framework. In Semantic Web Services: Theory, Tools and Applications, pages 281–316. IGI Global, 2007.
[23] Meditskos, G. and Bassiliades, N., 2009. Structural and role-oriented web service discovery with taxonomies in OWL-S. IEEE transactions on knowledge and data engineering, 22(2), pp.278-290.
[24] D. Hull, E. Zolin, A. Bovykin, I. Horrocks, U. Sattler, and R. Stevens. Deciding semantic matching of stateless services. In AAAI, pages 1319–1324, 2006.
[25] Bassiliades, N., Symeonidis, M., Meditskos, G., Kontopoulos, E., Gouvas, P. and Vlahavas, I., 2017. A semantic recommendation algorithm for the PaaSport platform-as-a-service marketplace. Expert Systems with Applications, 67, pp.203-227.
[26] T. Di Noia, E. Di Sciascio, F. M. Donini, and M. Mongiello. A system for principled matchmaking in an electronic marketplace. In Proceedings of the 12th international conference on World Wide Web, pages 321–330, 2003.
[27] D. Calvanese, M. Ortiz, M. Simkus, and G. Stefanoni. Reasoning about explanations for negative query answers in dl-lite. Journal of Artificial Intelligence Research, 48:635–669, 2013.
[28] Elsenbroich, C., Kutz, O. and Sattler, U., 2006, November. A Case for Abductive Reasoning over Ontologies. In OWLED (Vol. 216).
[29] P. Koopmann. Signature-based abduction with fresh individuals and complex concepts for description logics. In Description Logics, 2021.
[30] Ceylan, I., Lukasiewicz, T., Malizia, E., Molinaro, C. and Vaicenavicius, A., 2020. Explanations for negative query answers under existential rules.
[31] Del-Pinto, W. and Schmidt, R.A., 2019, July. ABox abduction via forgetting in ALC. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, No. 01, pp. 2768-2775).
[32] Du, J., Qi, G., Shen, Y.D. and Pan, J.Z., 2012. Towards practical ABox abduction in large description logic ontologies. International Journal on Semantic Web and Information Systems (IJSWIS), 8(2), pp.1-33.
[33] Du, J., Wang, K. and Shen, Y.D., 2014, June. A tractable approach to ABox abduction over description logic ontologies. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 28, No. 1).
[34] Halland, K. and Britz, K., 2012, October. ABox abduction in ALC using a DL tableau. In Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference (pp. 51-58).
[35] Pukancov´a, J. and Homola, M., 2017, July. Tableau-Based ABox Abduction for the ALCHO Description Logic. In Description Logics.
[36] Pukancová, J. and Homola, M., 2020. The AAA ABox Abduction Solver: System Description. KI-Künstliche Intelligenz, 34(4), pp.517-522.
[37] Klarman, S., Endriss, U. and Schlobach, S., 2011. ABox abduction in the description logic ALC. Journal of Automated Reasoning, 46(1), pp.43-80.
[38] Q. Ouyang, T. Dai, and Y. Ma. A hypergraph approach for logic-based abduction. DBKDA 2023, page 31, 2023.
[39] Bienvenu, M., 2008, September. Complexity of Abduction in the EL Family of Lightweight Description Logics. In KR (pp. 220-230).
[40] Baader, F., Küsters, R. and Molitor, R., 1999, July. Computing least common subsumers in description logics with existential restrictions. IJCAI (Vol. 99, pp. 96-101).
[41] Gocev, I., Meditskos, G. and Bassiliades, N., 2022, November. Towards Explaining DL Non-entailments by Utilizing Subtree Isomorphisms. In International Conference on Information Integration and Web (pp. 385-390). Cham: Springer Nature Switzerland.
[42] I. Gocev, S. Grimm and T. Runkler, "Supporting Skill-based Flexible Manufacturing with Symbolic AI Methods," IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society, Singapore, 2020, pp. 769-774, doi: 10.1109/IECON43393.2020.9254797.
[43] F. Baader, S. Brandt, and C. Lutz. Pushing the el envelope. pages 364– 369, 01 2005.
[44] Paolucci, M., Kawamura, T., Payne, T.R. and Sycara, K., 2002. Semantic matching of web services capabilities. In The Semantic Web—ISWC 2002: First International Semantic Web Conference Sardinia, Italy, June 9–12, 2002 Proceedings 1 (pp. 333-347). Springer Berlin Heidelberg.
[45] T. Di Noia, E. Di Sciascio, and F. M. Donini. Semantic matchmaking as non-monotonic reasoning: A description logic approach. Journal of Artificial Intelligence Research, 29:269–307, 2007.
[46] McGuinness, D.L., Shvaiko, P., Giunchiglia, F. and da Silva, P.P., 2004. Towards explaining semantic matching.
[47] S. Colucci, T. D. Noia, E. D. Sciascio, M. Mongiello, and F. M. Donini. Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace. In Proceedings of the 6th international conference on Electronic commerce, pages 41–50, 2004.
[48] T. Di Noia, E. Di Sciascio, and F. M. Donini. Extending semantic-based matchmaking via concept abduction and contraction. In Engineering Knowledge in the Age of the Semantic Web: 14th International Conference, EKAW 2004, Whittlebury Hall, UK, October 5-8, 2004. Proceedings 14, pages 307–320. Springer, 2004.
[49] Di Noia, T., Di Sciascio, E. and Donini, F.M., 2009, September. Computing information minimal match explanations for logic-based matchmaking. In 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (Vol. 2, pp. 411-418). IEEE.
[50] Di Noia, T., Di Sciascio, E., Donini, F.M. and Mongiello, M., 2003, August. Abductive matchmaking using description logics. In IJCAI (Vol. 3, pp. 337-342).

[51] Bartalos, P., 2011. Effective automatic dynamic semantic web service composition. Information Sciences and Technologies Bulletin of the ACM Slovakia, 3(1), pp.61-72.
[52] Rouached, M. and Godart, C., 2008, October. A Run-time service discovery tool for Web services compositions. In 2008 IEEE International Conference on e-Business Engineering (pp. 179-187). IEEE.
[53] Horridge, M., Parsia, B. and Sattler, U., 2008, October. Laconic and precise justifications in OWL. In International semantic web conference (pp. 323-338). Berlin, Heidelberg: Springer Berlin Heidelberg.
[54] Di Noia, T., Di Sciascio, E. and Donini, F.M., 2004. Extending semantic-based matchmaking via concept abduction and contraction. In Engineering Knowledge in the Age of the Semantic Web: 14th International Conference, EKAW 2004, Whittlebury Hall, UK, October 5-8, 2004. Proceedings 14 (pp. 307-320). Springer Berlin Heidelberg.
[55] Horridge, M. and Bechhofer, S., 2011. The owl api: A java api for owl ontologies. Semantic web, 2(1), pp.11-21.
[56] Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A. and Katz, Y., 2007. Pellet: A practical owl-dl reasoner. Journal of Web Semantics, 5(2), pp.51-53.
[57] Matentzoglu, N., Parsia, B.: Bioportal snapshot 30.03.2017 (Mar 2017). https://doi.org/10.5281/zenodo.439510
[58] Kindermann, C., George, A.M., Parsia, B. and Sattler, U., 2024, March. Minimal Macro-Based Rewritings of Formal Languages: Theory and Applications in Ontology Engineering (and Beyond). In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 38, No. 9, pp. 10581-10588).
[59] Hakeem, A., Sheikh, Y. and Shah, M., 2004, July. CASEE: a hierarchical event representation for the analysis of videos. In AAAI (pp. 263-268).
[60] Guo, Y., Pan, Z. and Heflin, J., 2005. LUBM: A benchmark for OWL knowledge base systems. Journal of Web Semantics, 3(2-3), pp.158-182.
[61] Shamir, R. and Tsur, D., 1999. Faster subtree isomorphism. Journal of Algorithms, 33(2), pp.267-280.
[62] Abboud, A., Backurs, A., Hansen, T.D., Vassilevska Williams, V. and Zamir, O., 2018. Subtree isomorphism revisited. ACM Transactions on Algorithms (TALG), 14(3), pp.1-23.

Appendix A. Functions

Here are additional functions used in the main implementation of our method.

A.1. equivClasses *function*

Algorithm 2 $equivClasses(x, N(x), T)$

$N(x)_{/\sim} \leftarrow \{\}$
for all $y \in N(x)$ **do**
  $[y] \leftarrow \{y\}$
  for all $z \in N(x)$ **do**
    if $\lambda_{\mathcal{E}}(\langle x, y\rangle) = \lambda_{\mathcal{E}}(\langle x, z\rangle)$ **then**
      $[y].add(z)$
    end if
  end for
  $N(x)_{/\sim} = N(x)_{/\sim} \cup ([y])$
end for
return $N(x)_{/\sim}$

A.2. μ function

Algorithm 3 $\mu(v, w, Nv_{/\sim}, Nw_{/\sim}, T_C, T_D)$

$\mu_{vw} \leftarrow \{\}$
for all $[x] \in N(v)_{/\sim}$ **do**
  for all $[y] \in N(w)_{/\sim}$ **do**
    if $\lambda_{\mathcal{E}_C}(\langle v, x\rangle) = \lambda_{\mathcal{E}_D}(\langle w, y\rangle)$ **then**
      $\mu_{vw} = \mu_{vw} \cup (\langle [x], [y]\rangle)$
    end if
  end for
end for
return μ_{vw}

A.3. constructHypotheses *function*

Algorithm 4 *constructHypotheses*$(\Phi, \sqsubseteq\sqsupseteq, T_C, T_D)$

$\mathcal{H} \leftarrow \{\}$
for all $\phi \in \Phi$ **do**
  for all $x \in \mathcal{V}_C \wedge x \notin \phi$ **do**
    $\lambda_{\mathcal{V}_C}(v) \leftarrow \bigsqcap \lambda_{\mathcal{V}_C}(v) \sqcap \bigsqcap_{\langle v,x\rangle \in \mathcal{E}_C} \exists \lambda_{\mathcal{E}_C}(\langle v,x\rangle).C_{T_C(v)}$
  end for
  for all $y \in \mathcal{V}_D \wedge y \notin \phi$ **do**
    $\lambda_{\mathcal{V}_D}(w) \leftarrow \bigsqcap \lambda_{\mathcal{V}_D}(w) \sqcap \bigsqcap_{\langle w,y\rangle \in \mathcal{E}_D} \exists \lambda_{\mathcal{E}_D}(\langle w,y\rangle).C_{T_D(w)}$
  end for
  $h \leftarrow \{\}$
  for all $\langle v,w\rangle \in \phi$ **do**
    $h.add(\bigsqcap \lambda_{\mathcal{V}_C}(v) \sqsubseteq\sqsupseteq \bigsqcap \lambda_{\mathcal{V}_D}(w))$
  end for
  $\mathcal{H}.add(h)$
end for
return $\mathcal{H}$